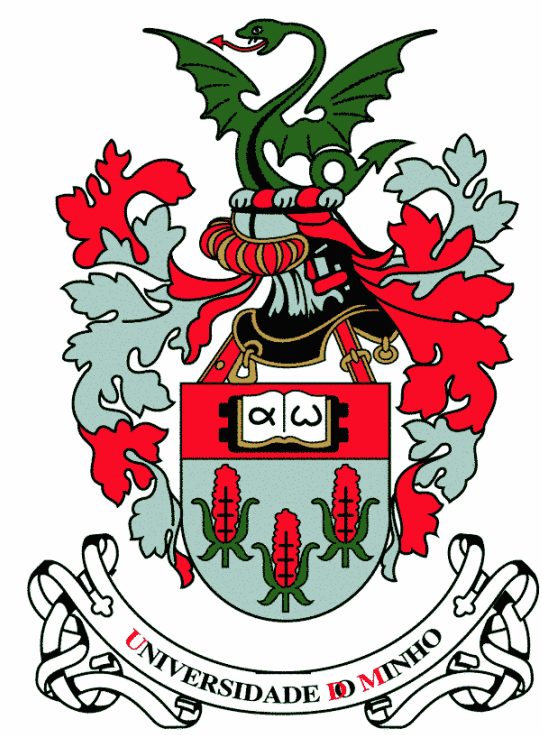
Web Services na Informação Geográfica

Mário André Araújo



Dissertação submetida à Universidade do Minho para obtenção do grau de Mestre em Informática, elaborada sob a orientação de Jorge Gustavo Rocha

Departamento de Informática
Escola de Engenharia
Universidade do Minho
Braga, 2005

Abstract

Open GIS Consortium has been promoting the multiplication of the Geo Web Services, especially thanks to the success of the Web Map Services and Web Feature Services. These services started giving the communication application-application a more operational use, allowing the exchange of information between clients and servers compliant with the standard, regardless of their internal specifications.

The scope of this work began with the study of the WS, namely the Geo Web Services in order to prove that only the syntatic interoperability[1] has been solved, i.e. the messages conform with the defined structure. Nevertheless, as it will be vindicated in this thesis, the interpretation and correct usage of the information requires a semantic analysis. The client's independent search and usage of Geo WS with relevant information is extremely difficult due to the lack of that semantic information.

It will be vindicated that a higher level of interoperability will be obtained by adding a semantic layer. Previous work on this topic, and two proposals are presented. One of the proposals is still in process and arises from an experiment on semantic interoperability carried by OpenGIS Consortium. The other proposal totally developed within the scope of this thesis, aims to create and maintain an ontology to add to the stack of Geo Web Services. This last approach is demonstrated by a tool developed to search GI in Geo Web Services, exploring a common multi-language ontology.

Therefore, our aim is to contribute to the improvement of the interoperability between Geo WS, i.e. allowing new IG servers to be replaced or added in a dynamic way. This dynamic behavior is necessary for example to enable mobile devices to discover and select servers with relevant information, while on movement.

[1] Capacity to communicate, execute programmes or data transfer between different functional units, without the user's concern about the specific characteristics of those units.(Rocha, 2004)

Resumo

A multiplicação dos Geo Web Services (Geo WS) tem vindo a ser conseguida sob os auspícios do consórcio OpenGIS, sendo de realçar o sucesso dos serviços *Web Map Service* e *Web Feature Service*. Estes serviços vieram operacionalizar a comunicação aplicação-aplicação, permitindo a troca de informação entre clientes e servidores, que respeitem a norma, independentemente das suas especificidades.

No âmbito deste trabalho, partiu-se do estudo dos WS e especificamente dos Geo WS para argumentar que apenas se resolveu o problema da interoperacionalidade[2] ao nível sintáctico, ou seja, as mensagens satisfazem a estrutura definida. No entanto, como se defende nesta tese, a interpretação e utilização correcta da informação carece de uma análise semântica. A procura e utilização autónomas de Geo WS com informação relevante, por parte de um cliente, são extremamente dificultadas pela falta dessa informação semântica.

Defende-se que, através da introdução de uma camada semântica se consegue atingir um nível superior de interoperacionalidade. Neste contexto, são identificados trabalhos já desenvolvidos nesta área e são apresentadas duas propostas para abordar esta problemática. Uma das propostas está ainda em consolidação e resulta de uma experiência de interoperacionalidade semântica do consórcio OpenGIS, em que a Universidade do Minho participa. Outra proposta, integralmente desenvolvida no âmbito deste mestrado, passa pela criação e manutenção de uma ontologia a adicionar à arquitectura dos WS. Esta última aproximação é demonstrada através de uma ferramenta desenvolvida para a procura de IG em Geo WS, sendo que os conceitos estão indexados a uma ontologia multi-língua comum.

Desta forma julgamos ter contribuído para melhorar a interoperacionalidade dos WS, isto é, possibilitando que dinamicamente se possam substituir ou acrescentar novos servidores de IG. Este dinamismo é necessário para que, por exemplo, um dispositivo móvel, à medida que se vai deslocando, possa encontrar e seleccionar servidores de informação relevante.

[2] Capacidade de comunicar, executar programas, ou transferir dados entre diferentes unidades funcionais sem que o utilizador tenha que se preocupar com as características específicas de cada uma dessas unidades (Rocha, 2004)

Agradecimentos

Um trabalho de mestrado tem o reconhecido significado a nível académico, mas representa acima de tudo, o culminar de um enorme esforço e dedicação, sobretudo para aqueles que o acumulam com a sua vida profissional. Este trabalho, em particular, não teria sido possível, sem o apoio de algumas pessoas que contribuíram de uma forma ou de outra para a sua realização.

Em primeiro lugar, tenho que agradecer ao meu orientador, Jorge Gustavo Rocha, pela dedicação, apoio, e sobretudo disponibilidade que sempre demonstrou mesmo a 400Km de distância.

Ao Professor Pedro Rangel Henriques, que, apesar da sua agenda complicada, me ajudou a arrancar com este projecto e garantir que ele seria sustentado até ao final.

Deixo também uma palavra de agradecimento ao Grupo de Sistemas de Informação Geográfica da Universidade do Minho, e em especial, ao meu ilustre colega Nuno Faria pelo seu discernimento e apoio para por as "coisas a funcionar".

Tenho que finalizar esta panóplia de agradecimentos académicos mencionando o Nuno Pereira e o Joel Vicente, pela força que me deram, muitas vezes sem saberem...

Quero também agradecer à minha família pelo apoio inequívoco que deu. Aos meus pais, irmãos e especialmente à minha mãe.

Tenho que deixar aqui uma palavra de agradecimento muito especial a todos os meus colegas da Siemens S.A. por suportarem, muitas vezes, a minha indisponibilidade, e em especial à Odete Cortiço, pelo seu incentivo e motivação.

À Iria, que mesmo no Qatar, se disponibilizou sempre para me ajudar.

Muito obrigado a todos!

Para a Sandra que esteve sempre comigo...

Conteúdo

Capítulo 1

Introdução

Com o aparecimento das redes informáticas e consequente massificação do uso da Internet, os computadores deixam de ser apenas máquinas que processam informação e assumem o papel de meio de comunicação. Através da Internet e com a utilização de um computador a informação pode ser trocada e partilhada por todos a um nível cada vez mais abrangente.

A evolução das tecnologias de telecomunicações e a procura pela mobilidade tiveram um impacto fundamental no aparecimento de novas plataformas e consequentemente novas necessidades de disponibilidade de informação. Pretende-se que a informação seja disponibilizada a qualquer hora, em qualquer lugar e para qualquer plataforma. A portabilidade da informação passa a ser um requisito essencial no âmbito da sua distribuição através da Internet. É neste contexto que aparece o XML (eXtensible Markup Language), que se pretende assumir como norma no intercâmbio de informação estruturada.

Ainda assim, a Web limita-se a ser um meio de interacção entre um humano e informação que aparece na forma de texto e gráficos. Por exemplo, diariamente podem ser consultadas informações de trânsito, estado do tempo, movimentos bancários, etc. Para este tipo de interacção a Web mostra-se eficiente, assumindo-se como o maior espaço para partilha de informação em suporte digital. No entanto, o humano aparece sempre como actor indispensável nesta cadeia de partilha e interpretação de informação.

Os WS são como que um passo em frente nesta interacção, ao permitirem não só a interacção de aplicações numa perspectiva aplicação-aplicação, mas também a automatização de tarefas sem a intervenção do ser humano.

A utilização de tecnologias baseadas em WS, e voltando novamente à necessidade já referida da portabilidade da informação, conjunta com a utilização de protocolos normalizados baseados em XML, proporciona a capacidade de intercâmbio dessa informação em ambientes eminentemente heterogéneos no contexto também já mencionado aplicação-aplicação.

Qual é então a relevância destas tecnologias no domínio da Informação Geográfica (IG) e dos sistemas de informação que a sustentam?

Há várias décadas que IG tem vindo a ser coleccionada em formato digital sendo que os

Sistemas de Informação Geográfica (SIG) são componente essencial na gestão, manutenção, representação, armazenamento, análise e transformação dessa IG. A complexidade da IG aliada à heterogeneidade de organizações, sistemas e processos que a manipulam, dificultam a partilha da mesma que acaba por ficar tendencialmente confinada ao contexto em que foi produzida. Estas características causam dificuldades no que diz respeito à partilha e acesso à IG em ambientes distribuídos e heterogéneos.

Porque as tecnologias SIG tendem a caminhar no sentido de adquirir capacidades que permitam a sistemas independentes trocar informação entre si de forma transparente, apesar de internamente a armazenarem em formatos distintos e potencialmente incompatíveis (Aguiar et al., 1998), os WS têm vindo a tornarem-se fortes aliados dos SIG proporcionando a estes capacidades de interacção – independentemente das suas especificidades.

Por isso, acreditamos que os WS podem ter um papel preponderante nesta questão e decidimos caminhar no sentido de investigar esta temática e explorar o potencial desta tecnologia no âmbito da pesquisa e partilha de IG.

Neste primeiro capítulo contextualiza-se o tema desta dissertação através de um conjunto de elementos introdutórios que passam pelo enquadramento deste trabalho com a realidade actual, pelas motivações que levaram ao estudo deste tema, os objectivos a que nos propostos e, por fim, a estrutura da dissertação.

1.1 Motivação

A disseminação dos Sistemas de Informação Geográfica (SIG) trouxe grandes expectativas sobre o impacto que estes sistemas podem vir a ter no nosso dia-a-dia, sendo que a massificação da utilização desses sistemas começa a ser uma realidade. Bons exemplos disso são as soluções que as três operadoras móveis existentes em Portugal oferecem no âmbito dos serviços baseados em localização, que de forma inequívoca mostra que existe de facto um interesse emergente no usufruto e disponibilização de serviços baseados em informação geo-referenciada.

É relativamente fácil extrapolar aplicações práticas que tornariam a nossa vida bem mais simples. Desde serviços informativos, como sendo informações de trânsito, locais de interesse, farmácias de serviço mais próximas, passando por utilizações lúdicas, como jogos capazes de tirar partido da localização dos jogadores, até aos sistemas de apoio ao cidadão implementados à escala local, regional ou nacional. São estes e outros exemplos que beneficiariam da utilização de uma infra-estrutura que permitisse a combinação de diversas fontes, independentemente da rede, plataforma, aplicação ou formato, ampliando assim o seu campo de acção.

No entanto, há que ter em conta que a Informação Geográfica (IG) é tradicionalmente um recurso proprietário e que existem muitas limitações na sua utilização. Essas dificuldades começam na localização da informação relevante, na escassa documentação dessa IG e na utilização de formatos proprietários que obrigam à aquisição de software específico para o seu processamento e manutenção.

1.2 Objectivos

A resposta a estas questões pode e deve ser facilitada pela tecnologia.

O enquadramento definido pelo consórcio OpenGIS passa por um conjunto de serviços disponibilizados na Web, através dos quais os clientes trocam IG, de acordo com formatos e protocolos bem definidos.

Estes serviços, permitem de facto que a IG seja trocada independentemente das suas especificidades. Assim está criado um nível de abstracção e são ultrapassadas as questões sintácticas inerentes. Por outro lado, até que ponto estes serviços podem ser concebidos para, autonomamente, procurarem e obterem informação relevante? Como se defende nesta disseração, esta autonomia só é possível com base na descrição semântica, quer dos serviços, quer da IG.

A motivação deste trabalho de mestrado assenta no estudo e desenvolvimento de soluções que permitam a utilização da IG no dia-a-dia, pelo utilizador comum. Os Geo Web Services são a área privilegiada de estudo desta dissertação. Em particular, enumeraram-se três objectivos no plano de trabalhos deste mestrado:

1. Conhecer em detalhe toda a tecnologia subjacente aos Geo WS. Isto porque os Geo WS despertam cada vez mais o interesse da comunidade dos utilizadores de IG e têm vindo a assumir um papel preponderante na sua disponibilização.
2. Perceber as limitações relativas à utilização dos Geo WS num ambiente em que o cliente necessita de descobrir e tirar partido de novos Geo WS.
3. Propor caminhos para sistematizar a descoberta e exploração de novos Geo WS.

1.3 Tese e Organização da Dissertação

A presente dissertação encontra-se organizada em 7 capítulos que, de certa forma, mostram a sequência que foi seguida ao longo deste trabalho, sequência esta que está delimitada pela Introdução (capítulo 1) e pela Conclusão (capítulo 7).

Os alicerces tecnológicos nos quais assenta este trabalho são os WS, pelo que estes são apresentados numa vertente técnica e pedagógica no capítulo 2. É feita uma apresentação dos conceitos fundamenteis que suportam esta tecnologia auxiliada por exemplos concretos de WS cujo objectivo passa por demonstrar a facilidade de desenvolvimento e instalação de WS num contexto Web. A interoperacionalidade entre componentes é também demonstrada recorrendo a serviços desenvolvidos em linguagens diferentes, neste caso Java e .NET.

Criar uma plataforma comum para partilha e disponibilização de dados e serviços baseados em IG que torne transparente a sua complexidade só é possível se houver uma base de entendimento capaz de orientar todos os actores na mesma direcção. Neste sentido, no capítulo 3 são apresentados alguns desses serviços preconizados pelo consórcio OpenGIS, com base no suporte tecnológico apresentado no capítulo anterior.

No capítulo 4 abordam-se conceitos relacionados com conhecimento, vocabulário, taxinomia, thesaurus e ontologia. Através de exemplos, discute-se a sua utilização no âmbito dos sistemas de informação em geral e nos SIG em particular.

Pretende-se, no capítulo 5, aprofundar o objecto de estudo desta dissertação e detalhar as motivações que lhe deram origem. Através de um caso de estudo demonstra-se o potencial dos Geo WS e as suas fraquezas.

Tendo por base o capítulo 5, abordam-se duas possíveis contribuições para a resolução do problema: (1) uma prova de conceito, em que se apresenta um motor de busca com a capacidade de fazer inferências semânticas sobre Geo WS, (2) detalha-se a visão do consórcio OpenGIS sobre estas questões, descrevendo uma experiência em curso no âmbito deste consórcio em que participa a Universidade do Minho.

Capítulo 2

Web Services

Os WS, no âmbito do tema que é abordado nesta dissertação, são a tecnologia que assume, conjuntamente com os Geo Web Services (capítulo 3), o maior protagonismo. Não seria possível estudar o seu papel no domínio da IG e dos SIG, sem conhecer bem essas duas soluções.

Pretende-se neste capítulo apresentar genericamente os WS e a sua arquitectura, não esquecendo a importância da normalização na procura pela interoperacionalidade e referindo as normas mais relevantes (SOAP, WSDL e UDDI). São apresentadas duas alternativas, suportadas por exemplos, para a criação de WS, a primeira utilizando Apache SOAP e a segunda recorrendo ao Apache Axis em que se demonstra a interacção entre um WS desenvolvido em Java e um cliente .NET.

2.1 Introdução

Os WS são aplicações Web com a capacidade para interagir entre si, permitindo a automatização de tarefas que só podiam ser feitas através da interacção dos humanos (Chung, 2003), ou seja, uma norma que define formas de interacção aplicação-aplicação recorrendo a formatos abertos. A utilização de protocolos normalizados, proporciona a capacidade de intercâmbio de informação entre aplicações em ambientes eminentemente heterogéneos.

Ao longo de vários anos outras iniciativas têm vindo a ser desenvolvidas no âmbito dos sistemas distribuídos resultando em diversos protocolos como DCOM (*Distributed Component Model*), CORBA (*Common Object Request Broker*) e J2EE (*Java 2 Platform, Enterprise Edition*). No entanto estas soluções são proprietárias, de difícil implementação e caras, o que, num ambiente heterogéneo como a Internet, as torna inadequadas. Os WS, por outro lado, aparecem como uma plataforma independente para suporte ao desenvolvimento de aplicações distribuídas sobre Internet respondendo a todas estas questões (Chung, 2003).

A ideia fundamental centra-se em tornar possível a criação de aplicações modulares

com capacidade de auto-descrição, para serem integradas na Internet e estarem acessíveis de e em toda a rede.

Sendo um WS uma aplicação vocacionada para comunicar com outras, torna-se necessária a definição de normas que possibilitem essa interacção.

2.1.1 Arquitectura dos Web Services

A arquitectura dos WS é apresentada de forma simplificada e ilustrada na figura 2.1. Esta prevê o envolvimento de três entidades distintas (Rocha, 2004):

O Fornecedor publica no catálogo, através da interface disponibilizada para o efeito, a existência de um novo serviço.

O Cliente após a consulta do catálogo, e se o serviço pretendido for encontrado, usufrui do mesmo directamente requisitando-o ao fornecedor.

O Catálogo informa o cliente fornecendo-lhe a descrição dos serviços e localização dos mesmos.

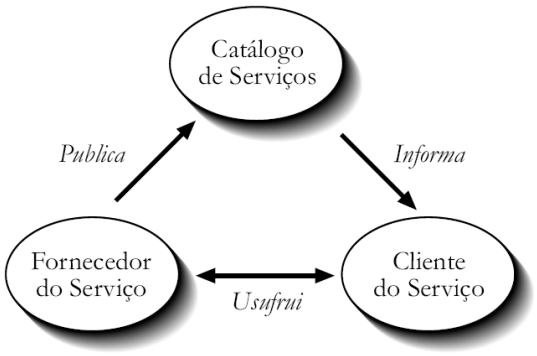


Figura 2.1: Arquitectura dos XML Web Services

Inicialmente, por parte da aplicação cliente, é enviada uma mensagem de maneira a descobrir o WS. Neste processo, a pedido do cliente, o catálogo fornece uma descrição das funcionalidades do serviço solicitado, informação essa que será utilizada por parte do cliente no sentido de este reconhecer as mensagens aceites pelo fornecedor do serviço. De salientar, que o catálogo não tem necessariamente conhecimento de todos os WS disponíveis, cabe assim ao WS a decisão de ser ou não publicado. Finalmente o serviço é invocado, através de troca de mensagens entre cliente e fornecedor.

2.1.2 Normalização

A desordem no desenvolvimento de WS põe em causa o seu grande objectivo – a interoperacionalidade – e mesmo apesar das recomendações do W3C (*World Wide Web Consortium*), os produtores de software começaram a seguir os seus próprios caminhos.

Em 2002, a *Web Services Interoperability Organization*[1] (WS-I), uma organização que reúne os maiores produtores de software, surge com o principal objectivo de criar uma "linha" orientadora para o desenvolvimento de WS de maneira a manter a interoperacionalidade, publicando para isso um documento, (Ballinger, 2003), que consiste numa série de recomendações a adoptar para o desenvolvimento de WS.

Só o tempo poderá dizer se estas recomendações serão de facto seguidas pelos produtores de software, ou se aparecerão outras. No entanto, o importante é que se chegue a um consenso para o desenvolvimento de WS.

A arquitectura apresentada na figura 2.1 necessita de normas que suportem a comunicação entre as três entidades mencionadas. Algumas dessas normas, definidas pelo W3C, são:

XML significa *eXtensible Markup Language*. É uma metalinguagem para anotação de documentos. Esta norma, (W3C, 1997), sob a orientação do W3C, define a sintaxe que a anotação deve seguir, ou seja, a forma como as anotações delimitam os elementos, a estrutura das anotações, etc. O facto de ser uma metalinguagem significa que não está limitada a um conjunto fixo de anotações, daí a sua extensibilidade. Sendo um documento XML constituído por texto, está garantida a independência de plataforma, pois com qualquer editor de texto se pode abrir o documento visualizando as anotações e os dados. Mais detalhes sobre esta norma e a as suas aplicações podem ser encontrados em (Ramalho and Henriques, 2002).

XML Namespaces recomendação do W3C publicada em (Bray, Hollander, and Layman, 1999). Um *namespace* é um conjunto de nomes, identificados por uma referência – URI (*Universal Resource Identifier*) – que são usados em documentos XML como prefixo dos nomes de elementos e atributos (Ramalho and Henriques, 2002). Esta recomendação define também como devem ser criados e declarados os *namespaces*.

XML Schema norma do W3C que descreve formalmente o conteúdo de um documento XML (Fallside, 2002). O XML Schema é também um documento XML. Estes permitem a definição da estrutura de documentos especificando os seus elementos e as suas relações, os tipos de dados, a existência ou não de atributos, etc.

SOAP *Simple Object Access Protocol*, protocolo de comunicação (Mitra, 2003), baseado em XML, que permite a troca de mensagens entre aplicações independentemente do protocolo de transporte utilizado. No entanto, no contexto dos WS, o protocolo de transporte utilizado é o HTTP.

WSDL *Web Service Definition Language*, como o próprio nome indica, trata-se de uma linguagem codificada em XML que permite fazer a descrição dos WS, em termos de métodos e respectivos parâmetros fornecidos pelo WS (Christensen et al., 2001).

UDDI *Universal Description, Discovery and Integration*, um serviço de registo e descoberta de WS descritos em WSDL, utilizado para armazenar, fornecer e devolver informação acerca de WS existentes (OASIS, 2004).

[1] `http://www.ws-i.org`

Seguidamente serão sumariamente descritas as normas que mais directamente influenciam os XML WS, são estas, o SOAP, a WSDL e o UDDI. Na sua tese de mestrado (Lopes, 2003), Lopes aborda em profundidade estas normas.

2.2 SOAP

O SOAP é uma linguagem XML que permite descrever as mensagens trocadas entre duas aplicações. Baseia-se em normas como o XML Schema e XML Namespace, que lhe confere independência em relação ao sistema operativo, linguagem de programação e aplicação.

Tendo como objectivo a troca de mensagens XML entre sistemas heterogéneos é necessário garantir que os intervenientes – o emissor e o receptor – consigam interpretar as mensagens. Para que isso seja possível é necessário definir também as regras e mecanismos que essa comunicação terá que respeitar. O SOAP é uma norma que especifica a estrutura dessas mensagens.

Ao estarem bem definidas, essas regras garantem que são conhecidos, pelas aplicações, os tipos de informação trocados e os detalhes de envio dessa mesma informação.

Como já foi dito anteriormente, o SOAP define a estrutura de um documento XML que as aplicações deverão utilizar para poderem comunicar entre si.

Tipicamente a comunicação é feita num só sentido, não existindo a noção de estado; as mensagens poderão ter de passar por diversos intermediários que as devem processar de alguma forma. Um **intermediário** é portanto uma entidade pela qual a mensagem vai passar antes de chegar ao **receptor**. O papel de um intermediário é esperar passivamente entre o emissor e o receptor até à chegada de uma mensagem, que este deve processar e reencaminhar para o respectivo destino (receptor). Ao conjunto de intermediários pelos quais a mensagem tem que passar desde o emissor até ao receptor chama-se **caminho**; a cada intermediário presente nesse caminho chama-se **actor**.

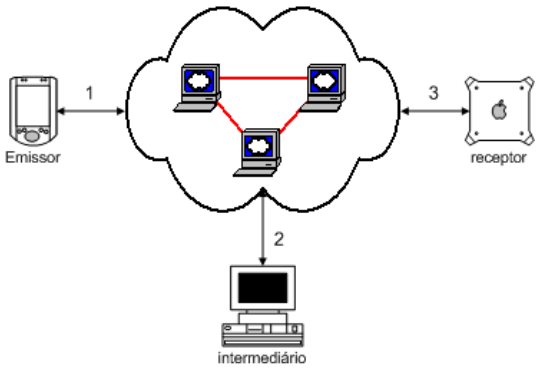


Figura 2.2: Intervenientes numa comunicação SOAP

O SOAP não especifica a construção do caminho entre os vários actores, existem no entanto protocolos de encaminhamento presentes em algumas implementações. A especifi-

cação menciona, no entanto, a forma de definir o papel de cada actor no processamento da mensagem. Este é sempre associado a blocos de cabeçalho e nunca ao corpo da mensagem.

Por exemplo, na figura 2.2, o emissor envia a mensagem para o receptor com a informação no cabeçalho, correspondente ao intermediário, que a processa removendo a parte que lhe é endereçada, e a reenvia para o receptor.

Ao longo desta secção falou-se em mensagens, em estrutura de mensagens, etc, no entanto é importante sistematizar toda esta informação. Para isso, recorrendo a exemplos simples, pretende-se mostrar como é constituída uma mensagem SOAP, que tipicamente é composta pelos seguintes elementos (ver figura 2.3):

Envelope este elemento é obrigatório e é o elemento principal de uma mensagem SOAP. Representa a raiz da mensagem SOAP e é especificado pelo elemento Envelope. O *namespace* associado a este elemento é `http://schemas.xmlsoap.org/soap/envelope`.

Cabeçalho (opcional) este elemento é o primeiro filho de Envelope. Uma mensagem pode conter zero ou mais blocos de cabeçalho, que serão embebidos no cabeçalho. Os atributo actor identifica o intermediário que deve processar o bloco de cabeçalho respectivo. Pode também conter o atributo mustUnderstand que é um atributo binário (pode conter apenas os valores *true* ou *false*). No caso de estar configurado com o valor *true*, obriga o receptor da mensagem a processar o cabeçalho que lhe é dirigido com sucesso, em caso contrário a mensagem deverá ser ignorada.

Corpo o corpo da mensagem é definido pelo elemento body. Trata-se de um elemento obrigatório e contém a informação da mensagem escrita em XML.

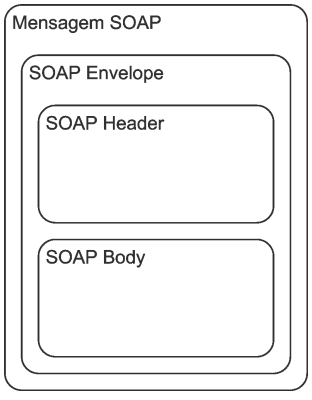


Figura 2.3: Estrutura de uma mensagem SOAP

Uma das formas mais comuns de utilização de SOAP é no contexto das *Remote Procedure Calls* (RPC). Neste sentido apresentam-se dois exemplos de mensagens SOAP, exemplos esses que têm como base o WS criado na secção 2.5, que serão percorridos de forma a salientar os aspectos mais importantes da estrutura da mensagem, mostrando-se de que forma são codificadas as operações, os seus atributos e valores de retorno.

O pedido SOAP seguinte, enviado através do protocolo HTTP, é feito sobre o serviço *CalculatorService*, identificado por urn:Calculator e cujo objectivo é o de calcular a soma de dois números – neste caso a soma entre 1 e 5.

Listagem 2.1: Exemplo de um pedido SOAP

```
1  POST /soap/servlet/rpcrouter HTTP/1.0
2  Host: 127.0.0.1:80
3  Content-Type: text/xml; charset=utf-8
4  Content-Length: 455
5  SOAPAction: ""
6
7  <?xml version='1.0' encoding='UTF-8'?>
8  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/
       " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http:
       //www.w3.org/2001/XMLSchema">
9  <SOAP-ENV:Body>
10 <ns1:sum xmlns:ns1="urn:Calculator" SOAP-ENV:encodingStyle="http://schemas.
       xmlsoap.org/soap/encoding/">
11 <v1 xsi:type="xsd:int">1</v1>
12 <v2 xsi:type="xsd:int">5</v2>
13 </ns1:sum>
14 </SOAP-ENV:Body>
15 </SOAP-ENV:Envelope>
```

Apesar de simples, este exemplo ilustra os principais elementos de uma mensagem SOAP: contém o elemento SOAP−ENV:Envelope, que é a raiz de todo o envelope e o elemento SOAP−ENV:BODY, filho do anterior. Os elementos opcionais SOAP−ENV:HEADER e SOAP−ENV:FAULT não estão presentes neste caso.

Analisando melhor o conteúdo do elemento SOAP−ENV:BODY verifica-se que este contém um documento XML bem formatado, em que todos os seus elementos são classificados com o *namespace* respectivo. Não podem existir referências a documentos DTD (*Document Type Definition)*[2] nem instruções de processamento.

O documento existente dentro do elemento SOAP−ENV:BODY tem como raiz o elemento ns1:sum, que representa uma chamada à operação de soma codificada no servidor – **public int** sum(**int** v1, **int** v2) – do WS CalculatorService.java. Os parâmetros da operação são passados nos elementos v1 e v2, estes elementos são do tipo **int** na linguagem Java que foi mapeado para o tipo xsd:**int** XML Schema. Note-se que o mapeamento de tipos de dados está dependente da implementação SOAP utilizada, no entanto, o importante a reter é que existem mecanismos que fazem esse trabalho.

Depois do pedido ser processado pelo servidor, o resultado da operação é codificado numa mensagem SOAP e retornado ao cliente. A resposta correspondente ao pedido efectuado na listagem 2.1 aparece de seguida, também transportada por HTTP.

[2]Documento escrito numa sintaxe formal que permite a descrição da estrutura de um documento XML (Ramalho and Henriques, 2002). É muito semelhante aos já mencionados XML Schemas, com algumas limitações, por exemplo, não é possível num DTD especificar o tipo ou formato dos dados.

Listagem 2.2: Resposta SOAP ao pedido da listagem 2.1

```
1  HTTP/1.1 200 OK
2  Set-Cookie: JSESSIONID=3B50D2E72C11C89EE410014D0F68C166; Path=/soap
3  Content-Type: text/xml;charset=utf-8
4  Content-Length: 450
5  Date: Sun, 19 Sep 2004 14:33:41 GMT
6  Server: Apache-Coyote/1.1
7  Connection: close
8
9  <?xml version='1.0' encoding='UTF-8'?>
10 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/
       " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http:
       //www.w3.org/2001/XMLSchema">
11 <SOAP-ENV:Body>
12 <ns1:sumResponse xmlns:ns1="urn:Calculator" SOAP-ENV:encodingStyle="http://
       schemas.xmlsoap.org/soap/encoding/">
13 <return xsi:type="xsd:int">6</return>
14 </ns1:sumResponse>
15
16 </SOAP-ENV:Body>
17 </SOAP-ENV:Envelope>
```

Esta mensagem, estruturalmente muito parecida com a anterior, no elemento SOAP-ENV:BODY contém o resultado da operação invocada – ns1:sum – no elemento ns1:sumResponse . O valor de retorno é do tipo Java **int** que, como já se mostrou anteriormente, corresponde ao tipo xsd:**int** em XML Schema, sendo que desta vez o mapeamento é feito no sentido inverso.

2.3 WSDL

A WSDL (*Web Services Description Language*) é uma linguagem XML para descrição de WS (Christensen et al., 2001).

Pode sumariar-se nos seguintes tópicos a informação contida num documento WSDL (Cerami, 2002):

- informação acerca dos métodos que serão publicados pelo serviço.
- os tipos de dados de entrada associados aos pedidos, e os tipos de dados de saída correspondentes às respostas.
- informação acerca do protocolo a ser utilizado para a invocação do serviço.
- endereço do serviço que deve ser utilizado pelo cliente aquando da invocação do mesmo.

Um documento WSDL não é mais do que um documento XML que obedece a uma estrutura fixa que permite definir a interface de um serviço.

Desta forma um qualquer cliente pode aceder ao conjunto de operações disponibilizadas por um WS de forma transparente no que diz respeito às tecnologias de desenvolvimento utilizadas.

Em WSDL existe uma separação clara entre a definição abstracta de portos e a sua concretização numa tecnologia específica. O mesmo se passa com a **mensagem**, onde primeiramente é feita a sua definição genérica para depois ser feita a tipificação da sua informação. Uma **operação** é um conjunto de uma mensagem de saída (pedido) e uma mensagem de entrada (resposta). Um **tipo de porto** é um conjunto de mensagens agrupadas por operações. O protocolo de transporte relativo a cada operação é especificado por uma **associação**. Um **porto** representa o endereço de rede do WS ao qual se pretende aceder, bem como o protocolo de transporte para o fazer; este é definido através de uma referência a uma associação. Por fim, um **serviço** é um conjunto de portos que são disponibilizados. A figura 2.4 pretende ilustrar os conceitos principais, em suma, uma operação, composta por duas mensagens – pedido e resposta – que por sua vez são compostas por *parts*, conceito este que será apresentado na secção seguinte.

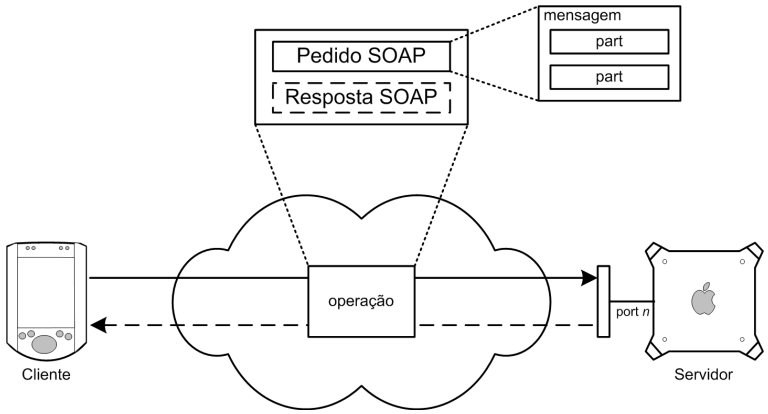


Figura 2.4: Mensagem WSDL

Sendo a WDSL uma linguagem XML, a sua estrutura é baseada em elementos e atributos, apresentando seguidamente os principais elementos desta linguagem (Chappell and Jewell, 2002) (ver figura 2.5):

definitions raiz do documento. É dentro deste elemento que são incluídas as propriedades que compõem o documento. Os *namespaces* que serão utilizados ao longo do documento são definidos neste elemento. Associado ao documento WSDL existe também um *namespace*, que é definido através do atributo targetNamespace neste elemento.

types este elemento define os tipos de dados que são utilizados ao longo do documento e que serão utilizados entre o servidor e o cliente durante a comunicação. Por omissão, os tipos de dados têm por base o XML Schema. No entanto, não é obrigatório que assim seja. Os tipos são definidos dentro deste elemento para que sejam posteriormente utilizados no elemento message.

message é neste elemento que se define a estrutura lógica e tipificada dos dados que são

transmitidos entre cliente e servidor. É importante salientar que a mensagem descrita neste elemento é num só sentido, ou seja, ou se trata de pedido ou de uma resposta, no entanto, um documento WSDL pode conter um ou mais elementos deste tipo. Os parâmetros do pedido ou valores de retorno da resposta são definidos através de um ou mais elementos do tipo part que podem ser incluídos como filhos deste elemento.

operation define uma operação suportada pelo serviço.

portType este elemento contém um conjunto de operações (definidas pelo elemento operation) suportadas por um serviço. A cada operação estão associados um pedido e uma resposta identificados respectivamente pelos elementos input e output.

binding associa as operações de um dado portType a uma tecnologia específica.

port um porto ou ponto de destino da rede definido através de uma associação, entre o elemento binding e um endereço de rede.

service conjunto de portos associados.

Um exemplo de um documento WSDL é apresentado na figura 2.6.

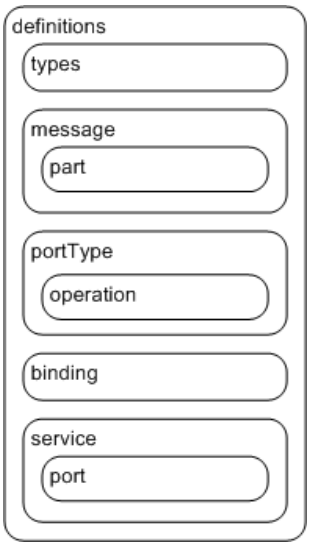


Figura 2.5: Anatomia de um documento WSDL

2.4 UDDI

O UDDI (*Universal, Description, Discovery and Integration*) trata-se de um projecto que tem como objectivo a criação de um protocolo para registar, publicar e descobrir WS num ambiente distribuído. Este projecto é liderado pelo consórcio OASIS[3] (*Organization*

[3] `http://www.oasis-open.org`

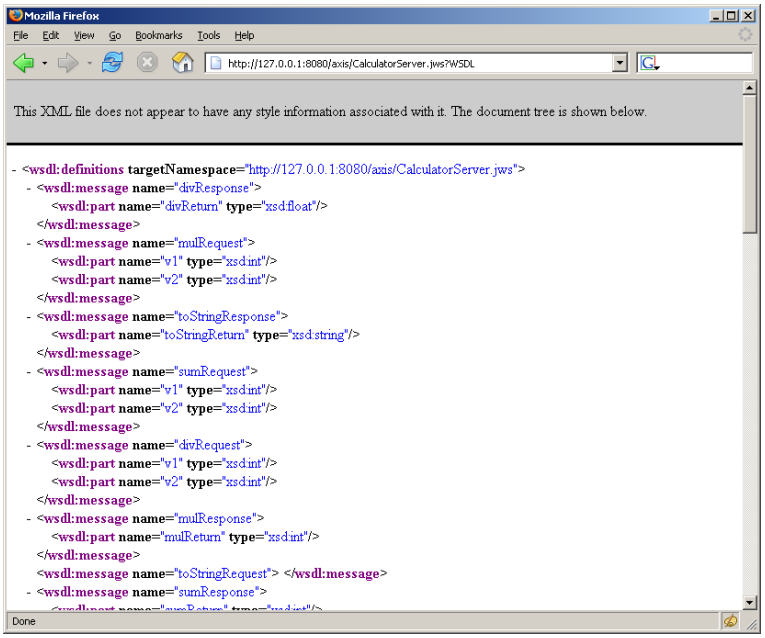


Figura 2.6: Exemplo de um documento escrito em WSDL

for the Advancement of Structured Information Standards). Este consórcio foi fundado pela IBM, Microsoft e Ariba, no entanto actualmente é composto por um conjunto mais abrangente de empresas (Bryan et al., 2002).

A especificação define um conjunto de WS que disponibilizam a descrição e descoberta de empresas ou outras organizações que forneçam WS, os WS que são disponibilizados por estas e por último as interfaces que podem ser utilizadas para gerir e aceder a esses WS (OASIS, 2004). Toda esta infra-estrutura é suportada por uma base de dados distribuída de WS.

Metaforicamente poder-se-ia dizer que o UDDI seria como que umas "páginas amarelas" de WS, na medida em que ambos são um registo de entidades, entidades estas que se apresentam com base nos dados que constam nesse registo. Também nas "páginas amarelas" se podem fazer pesquisas, eventualmente facilitadas com uma categorização da informação, e posterior invocação do serviço.

Conceptualmente existem três tipos de informação que podem ser consagrados no registo UDDI:

Páginas Amarelas organização dos WS em diversas categorias, como por exemplo entidade fornecedora, país de origem, entre outras. A pesquisa de WS neste contexto é feita com base nas já referidas categorias.

Páginas Brancas pretende-se que seja possível, através desta informação, pesquisar um WS com base em dados relativos à organização fornecedora do WS. Estes dados são constituídos por um conjunto de informações como sendo o nome, a morada, os

contactos, etc.

Páginas Verdes constitui um conjunto de informação técnica sobre o WS a ser disponibilizado. É com base nesta informação que o programador pode desenvolver uma aplicação que recorra ao WS em questão.

O modelo de dados do UDDI é definido por um XML Schema, que concretiza na prática os três tipos de informação que foram mencionados anteriormente:

businessEntity corresponde às páginas brancas contendo por isso informação relativa ao fornecedor do serviço.

businessService contém a informação que permite categorizar o WS, concretiza portanto o conceito já referido das páginas amarelas.

bindingTemplate informação técnica sobre o WS e a sua invocação, isto é, aquela informação classificada como fazendo parte das páginas verdes.

tModel utilizado para referenciar informação técnica adicional, como por exemplo um ficheiro externo.

Um fornecedor (businessEntity), entre outros dados, contém uma referência para uma lista de contactos e para uma lista de *Uniform Resource Locators* (URLs); estes URLs contêm um URI que aponta para um mecanismo alternativo de descoberta do WS. Contém também uma lista de businessServices. Por forma a descrever as diversas formas de invocação de um businessService, uma instância destes elementos referencia n bindingTemplates. A invocação de WS pode necessitar de informação adicional, essa é traduzida no modelo pelo elemento tModel que tem uma relação de *n-1* com o bindingTemplate.

2.5 Criação de um Web Service utilizando Apache SOAP

Nesta secção irá ser apresentada uma calculadora desenvolvida sob a forma de WS em Java recorrendo ao Apache SOAP. Pretende-se de uma forma simples e didáctica, mostrar como se desenvolve e instala um WS, quer do ponto de vista do cliente quer do ponto de vista do servidor.

O Apache SOAP[4] é uma implementação de SOAP desenvolvida pela *Apache Software Foundation*[5]. Esta implementação foi desenhada para correr num servidor HTTP que suporte *servlets*, sendo que ao longo deste exemplo será utilizado o Tomcat.

Para além do desenvolvimento das classes do WS propriamente dito, abordar-se-á também a instalação do WS no Tomcat e a configuração deste por forma a poder disponibilizar o WS desenvolvido.

[4] `http://xml.apache.org/soap`
[5] `http://www.apache.org`

Instalar e executar o Tomcat

Não se pretende abordar a instalação deste servidor, por esta razão apresentam-se apenas aqueles passos que são essenciais para a execução e instalação do WS apresentado nesta secção:

1. Para obter o binário para instalação é necessário aceder ao endereço oficial: `http://jakarta.apache.org/tomcat`.

2. Extrair o ficheiro jakarta−tomcat−5.0.27.zip. Assume-se que este foi extraído para ...\tomcat.

3. Adicionar ao ficheiro ...\tomcat\conf\tomcat−users.xml uma linha com o seguinte conteúdo: <user username="admin"password="admin"roles="standard,manager"/>. Só desta forma se pode ter acesso à consola de gestão do Tomcat – *Tomcat Manager* – que é necessária para a instalação do Apache SOAP.

4. Executar o ficheiro ...\tomcat\bin\startup.bat para iniciar o servidor.

5. Para testar a instalação, basta aceder ao endereço `http://127.0.0.1:8080/`, o resultado deve ser semelhante ao apresentado na figura 2.7.

Mais detalhes acerca da instalação e configuração do Tomcat podem ser obtidos em (Apache, 2004c).

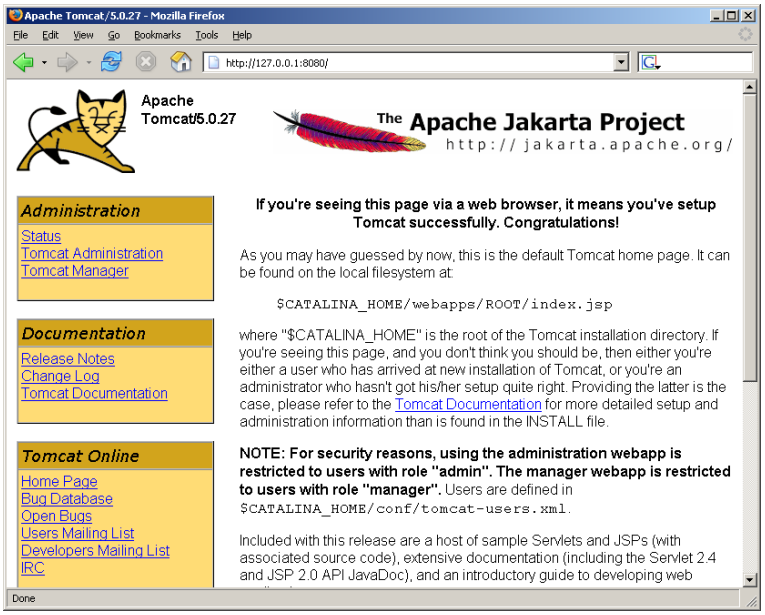


Figura 2.7: Página inicial de um servidor Tomcat

Instalar o Apache SOAP

O Apache SOAP pode ser instalado como servidor ou como cliente. Nesta secção serão abordadas ambas as possibilidades. Em qualquer dos casos a instalação resume-se à cópia de alguns ficheiros e à configuração da variável de ambiente CLASSPATH. De forma sucinta os procedimentos para a instalação são:

1. Instalação do Apache SOAP como servidor:
   (a) Extrair o ficheiro soap−bin−2.3.1.zip (pode ser obtido em `http://ws.apache.org/soap/`). Assume-se que este ficheiro foi extraído para o directório ..\soap.
   (b) Adicionar à CLASSPATH os seguintes seguintes ficheiros:
      i. soap.jar: este ficheiro pode ser obtido com a distribuição do Apache SOAP.
      ii. activation.jar: este ficheiro faz parte da *JavaBeans Activation Framework* e pode ser obtido em `http://java.sun.com/products/beans/glasgow/jaf.html`.
      iii. mail.jar: este ficheiro faz parte da *JavaMail* e pode ser obtido em `http://java.sun.com/products/javamail/`.
      iv. xerces.jar: este ficheiro faz parte da distribuição do Apache Xerces API para processamento de XML e pode ser obtido em `http://xml.apache.org/xerces-j`.
   (c) Para que todas as aplicações, e o próprio Tomcat, possam aceder às bibliotecas mencionadas anteriormente, é necessário copiar para ...\tomcat\shared\lib os ficheiros: activation.jar e mail.jar.
   (d) Através do *Tomcat Manager* e na opção *War File to Deploy*, incluir o ficheiro ...\soap\webapps\soap.war. O resultado desta operação é o mostrado na figura 2.8.
   (e) Para testar a instalação deve aceder-se aos seguintes endereços: `http://127.0.0.1:8080/soap/servlet/rpcrouter` e `http://127.0.0.1:8080/soap/servlet/messagerouter`, o resultado será a mensagem: "Sorry, I don't speak via HTTP GET- you have to use HTTP POST to talk to me".
2. Instalação como cliente: a instalação do cliente requer apenas que seja criada a variável de ambiente CLASSPATH conforme esta foi configurada anteriormente para o servidor.

Depois da instalação do Apache SOAP e do Tomcat, segue-se o desenvolvimento de todos os componentes do WS. Começando pelo servidor, o descritor e por fim o cliente.

Criar o Servidor

Como se pode ver na listagem 2.5, o código para a criação do servidor é muito simples, tratando-se apenas de uma classe sem qualquer especificidade inerente ao facto de se tratar de um WS. A classe CalculatorService contém os seguintes métodos:

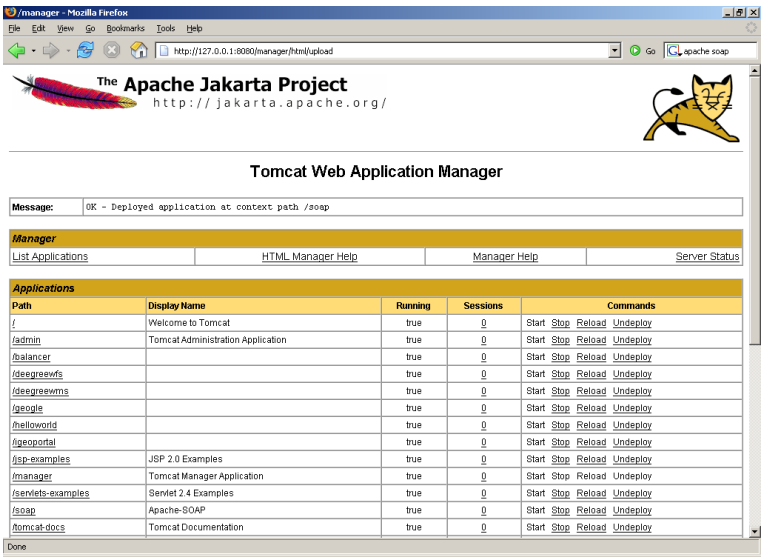


Figura 2.8: Resultado da Instalação do Apache SOAP no Tomcat

- sum: este método recebe dois inteiros e retorna a sua soma.
- mul: este método recebe dois inteiros e retorna o seu produto.
- div: este método recebe dois inteiros e retorna a sua divisão inteira (sem a parte decimal).

Listagem 2.3: CalculatorService.java

```
1  package samples.soap.calculator;
2
3  /**
4   * Classe que implementa uma Calucadora que efectua
5   * quatro operações.
6   *
7   * @author Mário André Araújo
8   */
9  public class CalculatorService {
10
11     public int sum(int v1, int v2){
12
13         return (v1 + v2);
14     }
15
16     public int mul(int v1, int v2){
17
18         return (v1 * v2);
19     }
20
21     public float div(int v1, int v2){
22
23         try {
```

```
24                  return (v1 / v2);
25          } catch(Exception e) {
26              return (0);
27          }
28      }
29  }
```

Criar o Descritor do Serviço

O Apache SOAP utiliza um documento XML com informação acerca do serviço, que pode ser gerado ou criado manualmente, a esse documento chama-se *Deployment Descriptor* (Apache, 2004a). Essa informação vai ser disponibilizada em tempo de execução ao servidor SOAP. Desta forma o servidor terá capacidade para, não só saber que serviços podem ser disponibilizados, mas também que operações podem ser feitas sobre esses serviços.

É através desta informação que o servidor vai poder fazer o reencaminhamento dos pedidos, neste caso HTTP, para a classe CalculatorService e o tratamento das respectivas respostas.

O descritor aparece seguidamente com os seguintes elementos principais: isd:service, isd:provider, isd: faultlistener e isd:mappings.

Listagem 2.4: Descritor do serviço apresentado na listagem 2.3

```
1  <isd:service
2     xmlns:isd="http://xml.apache.org/xml-soap/deployment"
3     id="urn:Calculator">
4
5    <isd:provider
6       type="java"
7       scope="Application"
8       methods="sum mul div">
9       <isd:java
10         class="samples.soap.calculator.CalculatorService"
11         static="false"/>
12      </isd:provider>
13
14      <isd:faultListener>
15         org.apache.soap.server.DOMFaultListener
16      </isd:faultListener>
17
18      <isd:mappings/>
19
20 </isd:service>
```

O elemento isd:service é a raiz do documento, neste exemplo, este elemento contém dois atributos:

- xmlns:isd: declaração do *namespace* correspondente.
- id: contém o URN do serviço, ou seja, no nome pelo qual o serviço vai ser identificado.

Seguidamente, e como primeiro filho de isd:service, aparece o elemento isd:provider, que contém os seguintes atributos:

- type: especifica que o serviço é desenvolvido em Java.
- scope: que indica o tempo de vida da instanciação da classe respectiva. O valor Application indica que o objecto existe até que o servidor seja terminado.
- methods: a lista de métodos a serem disponibilizados aos clientes do serviço. Esta lista é separada por espaços.

O elemento isd:java identifica a classe que implementa os métodos listados anteriormente. Para isso coloca-se o nome da classe no atributo **class**, note-se que é o nome completo da classe. É também necessário especificar se os métodos expostos são ou não estáticos, para isso utiliza-se o atributo binário **static**.

Criar o Cliente

O exemplo que se segue invoca o servidor apresentado anteriormente na listagem 2.3. Este código é bem mais complexo do que o apresentado para o servidor, pois contém toda a lógica de localização e invocação do serviço, no entanto o processo é sistemático e uma vez compreendido não deve criar grandes dificuldades.

Listagem 2.5: Calculator.java

```
1  package samples.soap.calculator;
2
3  import java.net.URL;
4  import java.util.Vector;
5  import org.apache.soap.Constants;
6  import org.apache.soap.Fault;
7  import org.apache.soap.rpc.Call;
8  import org.apache.soap.rpc.Parameter;
9  import org.apache.soap.rpc.Response;
10
11 /**
12  * Cliente do serviço CalculatorService.
13  *
14  * @author Mário André Araújo
15  * @see CalculatorService
16  */
17 public class Calculator {
18     private static final String SERVER_WS = "";
19     private static final String SERVER_URN =
20         "urn:Calculator";
21
22     public static void main(String[] args)
23         throws Exception{
24
25         URL url;
26         String methodName;
```

```
27          Call call;
28          Vector params;
29          Response resp;
30          Fault fault;
31
32          System.out.println("Chamando o servidor SOAP:\n");
33          url = new URL (args[0]);
34          methodName = args[1];
35
36          call = new Call();
37          call.setTargetObjectURI(SERVER_URN);
38          call.setMethodName(methodName);
39          call.setEncodingStyleURI(
40                  Constants.NS_URI_SOAP_ENC);
41          params = new Vector();
42
43          params.addElement (new Parameter("v1",
44                          Integer.class,
45                                  new Integer(args[2]), null));
46          params.addElement (new Parameter("v2",
47                          Integer.class,
48                                  new Integer(args[3]), null));
49
50          call.setParams (params);
51
52          System.out.print("\nO resultado e: ");
53          resp = call.invoke(url, SERVER_WS);
54
55          if (resp.generatedFault()){
56              fault = resp.getFault();
57              System.out.println (fault.getFaultString());
58          } else {
59              Parameter result = resp.getReturnValue();
60              System.out.print(result.getValue());
61              System.out.println();
62          }
63      }
64  }
```

Com base no diagrama de sequência apresentado na figura 2.9 será descrito o código apresentado na listagem 2.5:

1. Instanciação do objecto do tipo Call, que representa o pedido RPC que será feito.
2. Chamada ao método setTargetObjectURI com o parâmetro urn:Calculator indicando que o pedido será feito ao serviço identificado por esse parâmetro.
3. O nome do método do servidor a ser invocado é dado através do método setMethodName , que recebe como parâmetro o nome do método, neste exemplo a chamada foi feita ao método sum.
4. setEncodingStyleURI escolha do tipo de codificação a ser utilizado. No exemplo, é indicado o valor de NS_URI_SOAP_ENC representa o *namespace* `http://schemas.xmlsoap.org/soap/encoding`.

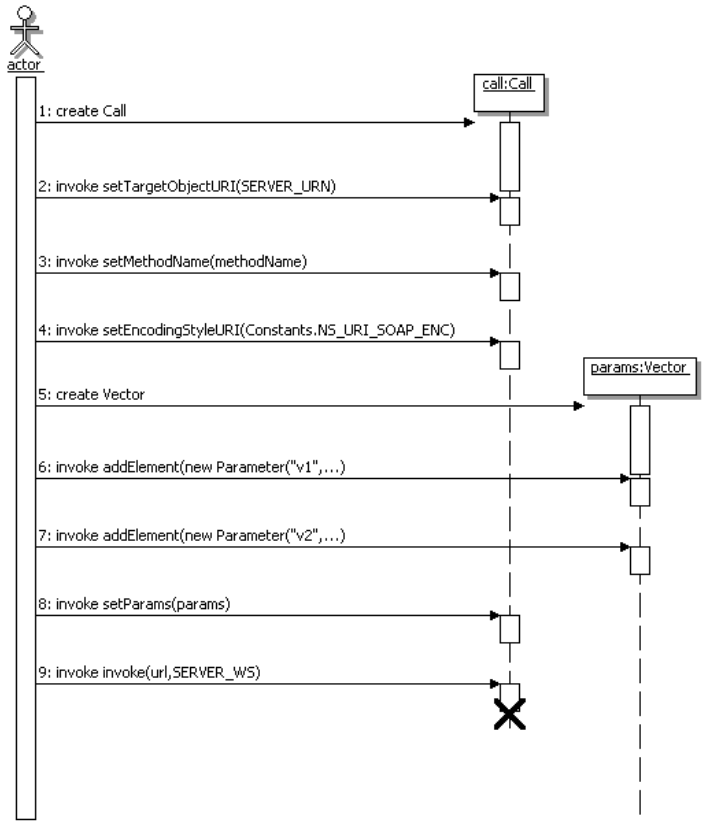


Figura 2.9: Diagrama de sequência do cliente

5. A chamada ao método sum implica a utilização dos dois parâmetros do mesmo, para isso é criado um Vector.

6. Ao primeiro parâmetro do método sum vai corresponder a primeira posição do vector params.

7. Ao segundo parâmetro do método sum vai corresponder a segunda posição do vector params.

8. Para associar o vector params ao objecto call faz-se uma chamada a setParams e passa-se como parâmetro o objecto params.

9. Finalmente, a invocação do método é feita efectuando uma chamada ao método invoke, passando o URL respectivo.

O resultado do pedido é passado para um objecto do tipo Response, que representa uma resposta a um pedido RPC. Em caso de falha, esta pode ser detectada através de uma chamada ao método generatedFault, caso contrário, pode obter-se o resultado da operação via método getReturnValue.

Instalação e execução do serviço

Após a criação do servidor, do cliente e do descritor e antes de se poder fazer chamadas ao serviço, é necessário instalá-lo no servidor HTTP. Para isso foi criado um ficheiro, baseado em *ant*[6], que sistematiza as regras de instalação e execução do serviço. A listagem é apresentada de seguida.

Listagem 2.6: build.xml

```
1  <?xml version="1.0"?>
2  <project name="build.soap.calculator" default="build">
3
4    <property file="build.properties"/>
5
6    <target name="build">
7      <javac destdir="${bin}" srcdir="${src}"
8        listfiles="true">
9        <classpath path="${ext}/soap.jar"/>
10     </javac>
11   </target>
12
13   <target name="clean">
14     <delete dir="${bin}/samples/soap/calculator"
15       verbose="true"/>
16   </target>
17
18   <target name="copy" depends="build">
19     <mkdir
20       dir="${TomcatSoapClasses}"/>
```

[6] http://ant.apache.org

```
21      <copy todir="${TomcatSoapClasses}/samples/soap"
22        verbose="yes">
23        <fileset dir="${bin}/samples/soap">
24          <include name="**/CalculatorService.class"/>
25        </fileset>
26      </copy>
27    </target>
28
29    <target name="deploy" depends="copy">
30      <java fork="true" dir="."
31       classname="org.apache.soap.server.ServiceManagerClient">
32       <classpath path="${ext}/soap.jar"/>
33       <classpath path="${ext}/mail.jar"/>
34       <classpath path="${ext}/activation.jar"/>
35       <classpath path="${ext}/xerces.jar"/>
36
37       <arg value=
38         "http://${ServerAddr}/soap/servlet/rpcrouter"/>
39       <arg value="deploy"/>
40       <arg value="ddCalculator.xml"/>
41      </java>
42    </target>
43
44    <target name="runSum">
45      <java classname="samples.soap.calculator.Calculator">
46        <classpath path="${bin}"/>
47        <classpath path="${ext}/soap.jar"/>
48        <classpath path="${ext}/mail.jar"/>
49        <classpath path="${ext}/activation.jar"/>
50        <classpath path="${ext}/xerces.jar"/>
51
52        <arg value=
53          "http://${ServerAddr}/soap/servlet/rpcrouter"/>
54        <arg value="sum"/>
55        <arg value="1"/>
56        <arg value="5"/>
57      </java>
58    </target>
59
60  </project>
```

A instalação do servidor é feita utilizando o comando ant deploy, este por sua vez executa a tarefa copy, que copia para o directório .../tomcat/webapps/soap/WEB−INF/classes o ficheiro CalculatorService.**class**. A execução do deploy propriamente dita é feita executando o ServiceManagerClient dando como parâmetros a instrução e o descritor.

Para validar a instalação pode recorrer-se à consola de administração do Apache SOAP, e verificar se a aplicação se encontra listada e em execução, conforme o ilustrado na figura 2.10.

A execução do cliente pode ser feita com o comando ant runSum, que vai fazer a chamada ao método sum com dois valores também passados por parâmetro. O resultado da execução deste comando é apresentado seguidamente:

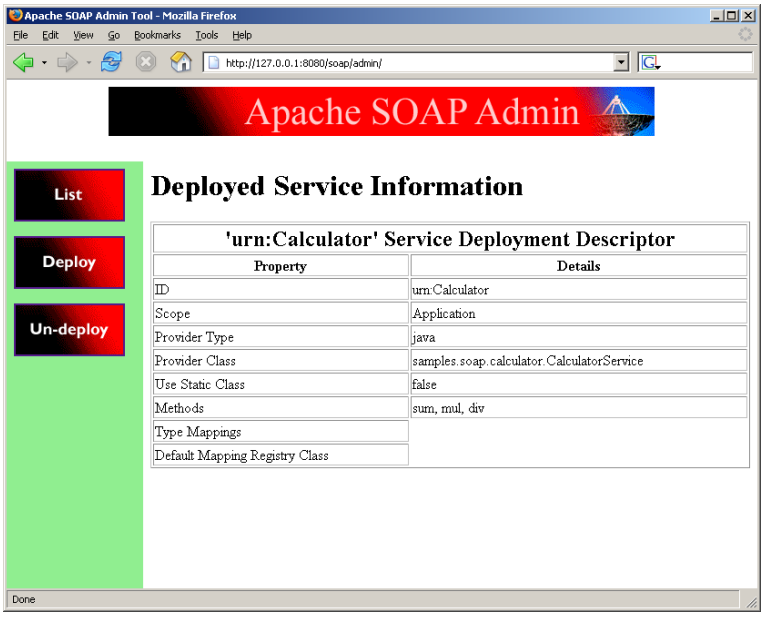


Figura 2.10: Descrição do WS fornecida pela consola de administração do Apache SOAP

```
1 Buildfile:
2 build.xml
3 runSum:
4      [java] Chamando o servidor SOAP:
5      [java] O resultado e: 6
6 BUILD SUCCESSFUL Total time: 734 milliseconds
```

Utilização do utilitário TCPTunnelGui

Juntamente com o Apache SOAP é fornecido um utilitário muito útil para inspeccionar a troca de mensagens SOAP entre o cliente e o servidor.

É possível ver as mensagens enviadas e recebidas através desta ferramenta que actua como um intermediário entre o cliente e o servidor Web onde reside o serviço.

Este utilitário pode ser executado através do comando seguinte (ver figura 2.11):

```
1 java org.apache.soap.util.net.TcpTunnelGui 80 127.0.0.1 8080
```

Ao actuar como um *Proxy* os pedidos SOAP feitos ao cliente passam pelo utilitário que os reencaminha para o servidor, o processo de recepção das respostas é o inverso. Os parâmetros utilizados para arrancar o utilitário são o porto que este deve escutar, o endereço IP do servidor SOAP e o porto para onde devem ser reencaminhadas as mensagens.

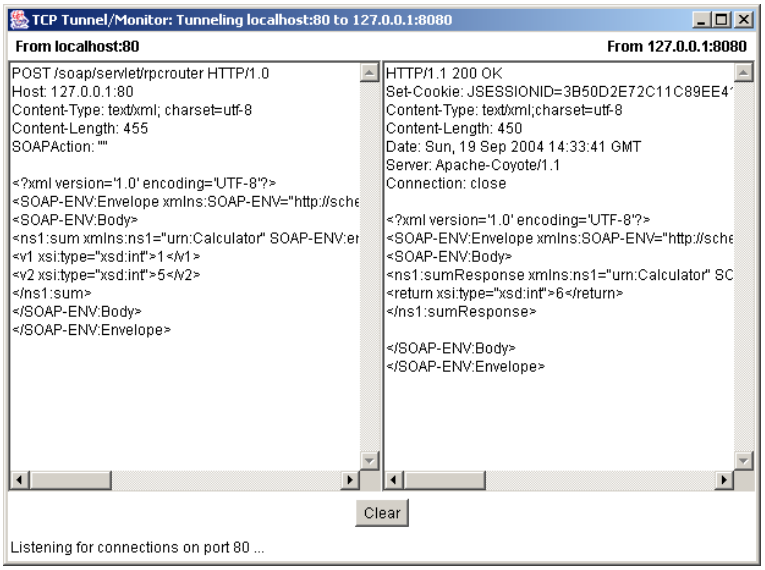


Figura 2.11: Exemplo de utilização do TCPTunnelGui

2.6 Web Services utilizando o Apache Axis

Após ter sido apresentado na secção anterior o processo de criação de um WS em Java recorrendo ao Apache SOAP, considera-se também relevante a apresentação de um exemplo utilizando uma outra tecnologia – o Apache Axis. Não só com o intuito de dar a conhecer a tecnologia propriamente dita, mas também de mostrar aquela que tanto tem sido apontada como a principal característica dos WS – a interoperacionalidade.

Para isto, com base no WS criado anteriormente, vai ser criado um outro WS sendo que desta feita utilizando o Apache Axis. No sentido de demonstrar a interoperacionalidade, são apresentados dois clientes desenvolvidos em linguagens de programação diferentes, Java e .NET, que acedem a esse WS.

O Apache Axis é uma implementação de SOAP baseada em Java que permite o desenvolvimento de WS nesta linguagem. Juntamente com esta implementação está incluído um ambiente de execução e uma API (*Application Program Interface*) que permitem o desenvolvimento de WS naqueles que são os seus componentes principais e com base em alguns protocolos normalizados.

O binário de instalação, que pode ser obtido no sítio `http://ws.apache.org/axis/`, inclui entre outras coisas (Nagappan, Skoczylas, and Sriganesh, 2003):

- Um ambiente de execução que pode ser utilizado como um servidor SOAP independente ou como aplicação integrada num servidor Web, como o Tomcat.
- Uma API Java para desenvolver aplicações e serviços baseados em SOAP.
- Infra-estrutura para criar WS sobre diversos protocolos de transporte.

- Ferramentas para converter WSDL em Java e vice-versa.
- Aplicações para instalar, monitorizar e testar WS.

Instalar o Apache Axis

A instalação do Apache Axis envolve a instalação de um servidor HTTP, neste exemplo pressupõe-se a existência do Tomcat cujo processo de instalação já foi mencionado na secção 2.5. A distribuição do Apache Axis inclui um directório webapps cujo conteúdo deve ser copiado para ...\ tomcat\webapps\axis. A partir desse momento pode testar-se a instalação acedendo ao endereço `http://127.0.0.1:8080/axis` (ver figura 2.12).

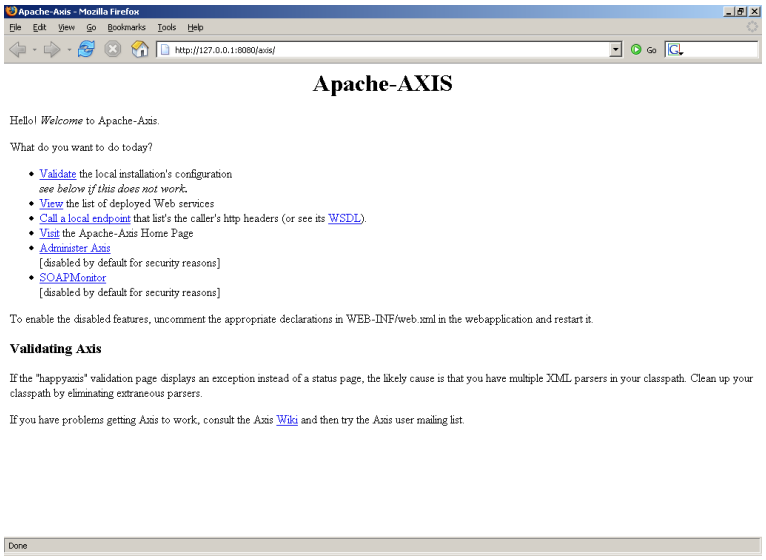


Figura 2.12: Verificação da instalação do Apache Axis no Tomcat

Existe também a possibilidade de executar o Apache Axis sem recorrer a nenhum servidor Web. Para isso, esta implementação inclui um servidor que pode ser executado através do comando seguinte:

```
1 java org.apache.axis.transport.http.SimpleAxisServer <porto>
```

Este servidor deve apenas ser utilizado apenas para testes e não em situações de distribuição.

Criação de um Cliente Java a partir de WSDL

Apresenta-se agora um exemplo cujo objectivo é demonstrar a utilização e papel do WSDL no desenvolvimento de WS. Por isso, em vez de desenvolver um WS, mostra-se como se desenvolve um cliente de um serviço já existente.

O processo para criar um cliente Java de um WS está ilustrado na figura 2.13, e consiste no seguinte:

1. Inquirir o serviço Axis no sentido de obter a descrição WSDL para um dado WS.
   (a) Este por sua vez, inspecciona o WS em questão, e retorna o documento WSDL correspondente.
2. Com base no documento WSDL previamente retornado, são geradas as classes que irão fornecer o acesso ao serviço de forma transparente – *Proxies*.
3. Agora, recorrendo às classes geradas anteriormente pode obter-se uma referência para o WS.
4. Através da referência do serviço que foi obtida, consegue-se invocar os métodos por este disponibilizados.

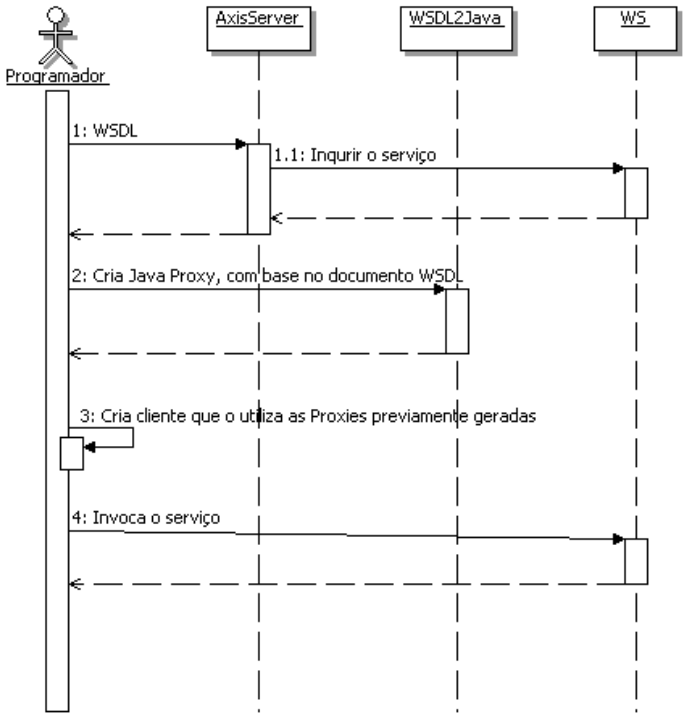


Figura 2.13: Criação de um cliente SOAP utilizando Apache Axis

Para inquirir o WS, no sentido de obter a sua descrição WSDL, pode utilizar-se a sequência de comandos *ant* que se segue:

Listagem 2.7: Inquirir a descrição WSDL do serviço

```
73  <!-- Get WSDL from server -->
74  <target name="get.calculator.wsdl" depends="deploy.adminclient">
75    <get src="${axis.url}/services/CalculatorService?wsdl"
76      dest="${gen}/CalculatorServer.wsdl"/>
77  </target>
```

O resultado é a criação de um ficheiro CalculatorServer.wsdl, este pode ser aberto num qualquer editor de texto, podendo-se facilmente identificar que operações tem este WS publicadas. No entanto, o objectivo não é somente obter a descrição do serviço, mas sim, com base nesta, criar uma infra-estrutura que permita o acesso ao serviço de forma simples. Neste sentido a sequência de comandos *ant* que se segue cumpre esse propósito.

Listagem 2.8: Geração das *proxies* com base da descrição WSDL do serviço

```
79  <!-- Generate CalculatorServer proxy classes -->
80  <target name="gen.calculator.java">
81   <axis-wsdl2java url="${axis.url}/services/CalculatorService?wsdl"
82    output="${gen}">
83   </axis-wsdl2java>
84  </target>
```

Após a execução deste comando são criados os seguintes ficheiros (Apache, 2004b):

- CalculatorServer.java: uma interface Java com os métodos disponibilizados pelo servidor. A geração de uma interface deste tipo é feita por cada elemento portType existente na descrição WSDL do serviço.
- CalculatorServerService.java e CalculatorServerServiceLocator.java: porque um cliente não deve instanciar um *stub* directamente, são também geradas estas duas classes. Um *service locator* é instanciado, e através de um método para o efeito, o cliente obtém uma referência para o *stub*.
- CalculatorServiceSoapBindingStub.java: um *stub* que contém o código que converte as invocações dos métodos em chamadas SOAP para o servidor. Funciona como um *proxy* eliminando a lógica de invocação remota entre cliente e servidor, ou seja, o cliente pode invocar métodos remotos do servidor como se de métodos locais se tratassem.

Com a ajuda das classes criadas através do comando axis-wsdl2java, a invocação do serviço processa-se conforme o exemplificado seguidamente:

Listagem 2.9: Cliente Java de um WS utilizando o Axis

```
1  package samples.axis.calculator;
2
3  import java.rmi.RemoteException;
4
5  import javax.xml.rpc.ServiceException;
6  import _1._0._0._127.axis.services.CalculatorService.CalculatorServer;
7  import _1._0._0._127.axis.services.CalculatorService.CalculatorServerService
       ;
8  import _1._0._0._127.axis.services.CalculatorService.
       CalculatorServerServiceLocator;
9
10
11
12 public class CalculatorClient {
13
```

```
14
15
16          public static void main(String[] args) {
17
18                  //lookup
19                  CalculatorServerService service = new
                        CalculatorServerServiceLocator();
20                  try {
21                          //get service
22                          CalculatorServer server = service.
                                getCalculatorService();
23                          //invoke service
24                          System.out.println("4 / 2 = " + server.div(4, 2));
25                          System.out.println("4 + 2 = " + server.sum(4, 2));
26                          System.out.println("4 x 2 = " + server.mul(4, 2));
27
28                  } catch (ServiceException e) {
29                          e.printStackTrace();
30                  } catch (RemoteException e) {
31                          e.printStackTrace();
32                  }
33          }
34  }
```

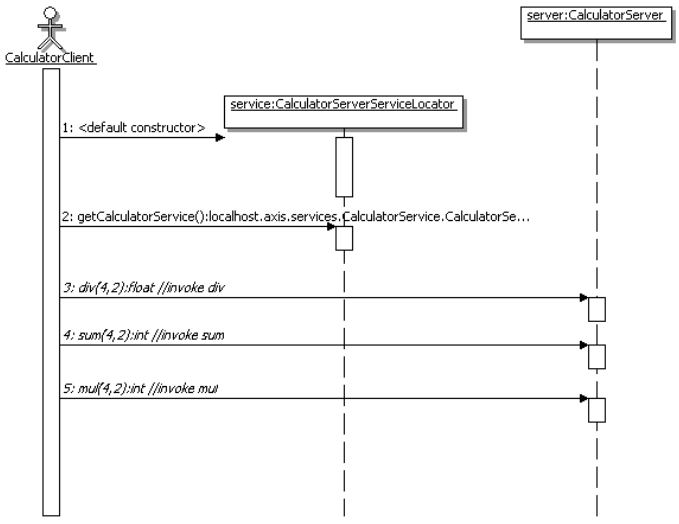


Figura 2.14: Invocação de um WS Axis

O cliente (ver figura 2.14) começa por instanciar o *service locator*. Posteriormente utiliza o método getCalculatorService para obter uma referência do *stub* do serviço. Depois as chamadas ao WS são feitas através dos métodos fornecidos pelo *stub* – div, mul e add.

Cliente de um cliente .NET a partir de WSDL

Da mesma forma que se cria um cliente Java para um WS desenvolvido através do Apache Axis, pode criar-se um cliente .NET. Isto mostra a interoperacionalidade entre WS e atesta a sua capacidade de interacção em ambientes heterogéneos.

À semelhança do que acontece com o cliente Java que se mostrou anteriormente também é necessário que, através do WSDL do serviço, sejam gerados os *stubs* que vão ser utilizados pelo cliente para se ligar ao servidor.

Para isso utiliza-se a tarefa WsdlToDotnet conforme mostra a listagem seguinte:

Listagem 2.10: Criação dos stubs em C#

```
104  <target name="cs.calculator">
105
106    <copy file="${src}/${pkg.calculator}/calculator.cs"
107      tofile="${gen}/Calculator.cs"/>
108
109    <WsdlToDotnet destfile="${gen}/CalculatorServerService.cs"
110      url="${axis.url}/services/CalculatorService?wsdl"/>
111
112    <csc srcDir="${gen}"
113      destfile="${gen}/client.exe" targetType="exe"/>
114
115  </target>
```

A execução desta tarefa vai gerar um *stub*, que depois de instanciado permite a invocação transparente dos métodos do servidor. Pelo facto de ser gerado somente um ficheiro, este exemplo é bem mais simples que a alternativa Java. A listagem seguinte contém o código do cliente:

Listagem 2.11: Cliente C#

```
1  using System;
2
3  public class Calculator {
4
5          public static void Main(String[] args) {
6
7                  CalculatorServerService service = new
                       CalculatorServerService();
8
9                  Console.WriteLine("2 / 2 = " + service.div(2,2));
10                 Console.WriteLine("2 + 2 = " + service.sum(2,2));
11                 Console.WriteLine("2 x 2 = " + service.mul(2,2));
12         }
13 }
```

A lógica da invocação do serviço está totalmente encapsulada pela classe CalculatorService , o que faz com que a chamada aos métodos do serviço seja feita como se este fosse local.

Como se queria provar, foi possível desenvolver programas baseados em tecnologias diferentes que trocam dados entre si, seguindo um conjunto de normas já estabelecidas.

No próximo capítulo mostra-se que esta interoperacionalidade é fundamental para compor programas que manipulem informação geográfica.

Capítulo 3

Geo Web Services

Apresentados os WS, no capítulo 2, abordam-se os Geo Web Services que são uma proposta do consórcio OpenGIS, anteriores à normalização dos XML WS, não sendo suportados por SOAP, WSDL e UDDI. Concentra-se o nosso estudo em dois dos serviços preconizados pelo consórcio, o *Web Mapping Service* (WMS) e o *Web Feature Service* (WFS) sobre os quais se descreve a norma e se exemplifica a sua utilização.

3.1 Introdução

O aparecimento dos WS trouxe grandes expectativas no âmbito da sua utilização como forma de valorização da IG em diversas vertentes. Aos WS podem ser confiadas diversas funções, desde a localização de IG relevante até ao encapsulamento de especificidades dessa IG, como sendo por exemplo o sistema de coordenadas em que esta se encontra. Uma valência que pode vir a ser conseguida, tem a ver com a partilha da informação de forma cooperativa, isto é, aproveitando a bidireccionalidade que os WS oferecem podem criar-se repositórios que permitam não só operações de consulta mas também de actualização da IG.

O consórcio OpenGIS[1], uma associação sem fins lucrativos, dedicada a promover novas abordagens, técnicas e comerciais, para a interoperacionalidade tem como objectivo primordial resolver os problemas de partilha de dados e transferir os SIG do seu ambiente actual, suportados maioritariamente por ambientes fechados e monolíticos, para ambientes suportados por tecnologias abertas e distribuídas.

Como documento orientador para o desenvolvimento de serviços que promovam a interoperacionalidade entre fontes de informação e tecnologias surge o *OpenGIS Reference Model* (ORM) (Buehler et al., 2002) que apresenta as linhas orientadoras para a criação de uma plataforma de trabalho, sobre a qual as empresas de software podem desenvolver aplicações que valorizem a interoperacionalidade da IG.

O desejo da criação de um ambiente distribuído que promova a já mencionada interope-

[1] `http://www.opengis.org`

racionalidade entre tecnologias e fontes de informação começa antes mesmo da existência dos WS (OpenGIS, 1998).

Os XML Web Services correspondem a um esforço de normalização posterior aos OpenGIS Web Services. Por isso, os Geo-WS não contemplam a auto-descrição através de um documento WSDL, nem os mecanismos de catalogação UDDI e também não suportam o encapsulamento das mensagem em SOAP.

As mensagens são trocadas no protocolo HTTP (através das operações GET e POST), com a vantagem de se poderem testar usando um navegador vulgar, como o Firefox. A auto-descrição é feita através de uma operação comum a todos os Geo-WS, GetCapabilities, que devolve um documento XML. Não existe a noção de catálogo para registo dos Geo-WS disponibilizados, semelhante ao UDDI.

Numa tentativa de criar um modelo conceptual que defina um conjunto de serviços que abranja todas as áreas da IG, o consórcio OpenGIS sugere a criação de um conjunto de WS, através dos quais os clientes trocam IG, de acordo com formatos e protocolos bem definidos. Alguns desses serviços já gozam de um certo consenso, mas outros ainda estão em discussão. Neste contexto serão apresentados de forma detalhada dois serviços:

- *Web Mapping Service*
- *Web Feature Service*

A descrição dos serviços será auxiliada por exemplos baseados no projecto deegree [2], que fornece implementações de referência para serviços especificados pelo OpenGIS.

3.2 Invocação dos Serviços

Cada um dos serviços mencionados anteriormente – WMS e WFS – fornece uma série de operações para serem invocadas pelos seus clientes. Estas operações podem ser executadas através a utilização de um navegador comum sob a forma de URL – utilizando o método GET. Nos exemplos apresentados ao logo deste capítulo recorreu-se ao método POST quando a implementação do serviço utilizada não suportava o método GET.

Um pedido efectuado utilizando o método GET é um URL que contém o endereço HTTP ou HTTPS do servidor, seguido de um conjunto de pares nome=valor que serão passados à operação a ser invocada. A sintaxe de um pedido válido é dada pela expressão http://*host[:porto]/caminho[?{nome[=valor]&}]*, onde http://*host[:porto]* representa nome ou IP do servidor que aloja o serviço seguido do porto (opcional). O caminho representa a estrutura de directorias, dentro do servidor, que aponta para o serviço. Os parâmetros de entrada que são enviados para o servidor aparecem depois do símbolo ? no formato nome=valor, sendo que se existir mais do que um par, o símbolo & tem que ser utilizado como separador.

Existem três parâmetros que são comuns na invocação de qualquer operação num dos serviços apresentados:

[2] `http://www.deegree.org`

VERSION este parâmetro especifica a versão do protocolo que o cliente espera que seja suportada pelo servidor. Um servidor pode suportar mais do que uma versão do mesmo serviço, sendo que o cliente pode negociar com o servidor a versão que ambos querem utilizar.

SERVICE este parâmetro é obrigatório e indica qual o serviço a ser invocado no pedido. Por exemplo, para o caso de um pedido a um WMS o valor a ser utilizado é "WMS".

REQUEST a operação a ser invocada. Trata-se também de um parâmetro obrigatório. Por exemplo, para o caso da descrição do serviço deve ser utilizado o valor "GetCapabilities".

3.3 Web Mapping Service

O Web Mapping Service (WMS) normaliza o processo de requerer mapas por parte do cliente. Os mapas são requeridos a um WMS, identificando um conjunto de camadas e respectivos parâmetros necessários à sua visualização.

Uma funcionalidade interessante do WMS é a capacidade de combinar diferentes serviços, ou seja, é possível compor um mapa com dados provenientes de diferentes e distribuídos WMSs.

Na especificação destes serviços, (OpenGIS, 2002b), são definidas três operações (ver figura 3.1):

GetCapabilities que retorna meta-informação do serviço, descrevendo-o e enumerando os parâmetros que este aceita – secção 3.3.1.

GetMap que retorna uma imagem do mapa de acordo com os parâmetros especificados no pedido – secção 3.3.2.

GetFeatureInfo (opcional) retorna informação acerca de entidades específicas apresentadas no mapa – secção 3.3.3.

3.3.1 GetCapabilities

O objectivo da operação GetCapabilities é o retorno de meta-informação do serviço. Informação esta, que é fornecida num formato que deve ser interpretado por humanos e pelo computador, daí a escolha do XML como protocolo a ser utilizado.

Um **pedido** GetCapabilities é composto pelos seguintes parâmetros:

VERSION (opcional) versão do serviço a utilizar.

SERVICE no caso de um pedido a um WMS o valor a ser utilizado é "WMS".

REQUEST No caso a operação GetCapabilities deve ser utilizado o valor "GetCapabilities".

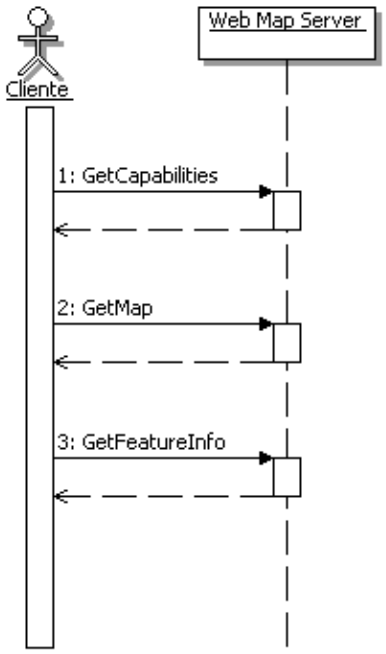


Figura 3.1: WMS – Operações disponibilizadas

FORMAT (opcional) indica o formato do retorno da operação. Ao tratar-se de um parâmetro opcional, se for omitido, o servidor responde em XML (text/xml), este formato vai ser também utilizado no caso de ser introduzido um formato que não é suportado pelo servidor.

UPDATESEQUENCE (opcional) utilizado para que ambos – cliente e servidor – mantenham um estado acerca dos dados retornados pela operação (*cache*). Desta forma o cliente sabe que foram feitas alterações à meta-informação do serviço desde a última chamada que efectuou.

O URL seguinte é um exemplo de um **pedido** GetCapabilities a um WMS:
http://www.mapsherpa.com/cgi-bin/wms_iodra?SERVICE=WMS&VERSION=1.1.1
&REQUEST=getcapabilities&FORMAT=text/xml

Neste pedido pode ver-se a invocação a um WMS (SERVICE=WMS) utilizando a versão "1.1.1" (VERSION=1.1.1) executando a operação GetCapabilities (REQUEST=GetCapabilities) cujo retorno deverá ser feito em XML (FORMAT=text/xml).

A **resposta** a uma invocação do tipo GetCapabilities é um documento XML com a meta-informação do serviço em questão cuja estrutura é especificada pelo OpenGIS através de um DTD [3].

[3] `http://schemas.opengis.net/wms/1.1.1/WMS_MS_Capabilities.dtd`

A raiz do documento retornado é o elemento WMS_MS_Capabilities (ver figura 3.2), documento este que tem duas componentes de informação fundamentais: (a) meta-informação sobre o serviço, (b) meta-informação sobre as operações suportadas e, opcionalmente, as camadas do mapa disponibilizadas pelo servidor.

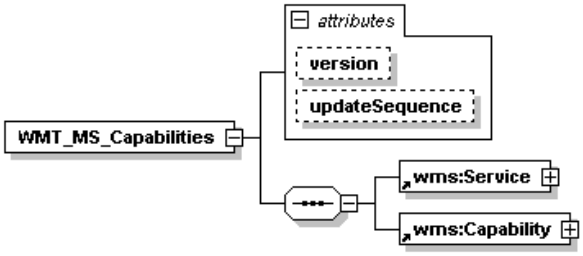


Figura 3.2: WMS – Estrutura do elemento WMS_MS_Capabilities

A meta-informação do serviço (ver figura 3.3), que está contida dentro do elemento Service, contém dados gerais sobre o serviço. Por sua vez, estes dados são incluídos através dos elementos:

Name neste caso, que se trata de um WMS, o nome tem que ser OGC: WMS.

Title uma descrição curta do serviço, para ser utilizada para efeitos de listagem apresentada a um utilizador.

Abstract trata-se também de uma descrição, um pouco mais extensa que a anterior, e num formato narrativo.

KeywordList lista de palavras-chave que permitam ao serviço ser catalogado e pesquisado através dessas palavras-chave.

OnlineResource pode ser utilizado, por exemplo, para referenciar o sítio do fornecedor deste serviço.

ContactInformation (opcional) este elemento deve conter informação de contacto de uma pessoa ou entidade responsável por dar suporte ao serviço.

Fees (opcional) se o serviço tiver algum tipo de taxa associada à sua utilização, essa informação deve ser incluída aqui. No entanto, no caso de não existir qualquer taxa, deve utilizar-se a palavra reservada none (<Fees>none</Fees>).

AccessConstraints (opcional) no caso do serviço estar sujeito a restrições de acesso ou utilização, pode usar-se este elemento. De salientar que não existe qualquer tipo de sintaxe definida para este efeito. No caso de não existir qualquer restrição, deve utilizar-se a palavra reservada none (<AccessConstraints>none</AccessConstraints>).

A listagem de um extracto do documento XML que contém a meta-informação acerca de um serviço é aqui apresentada:

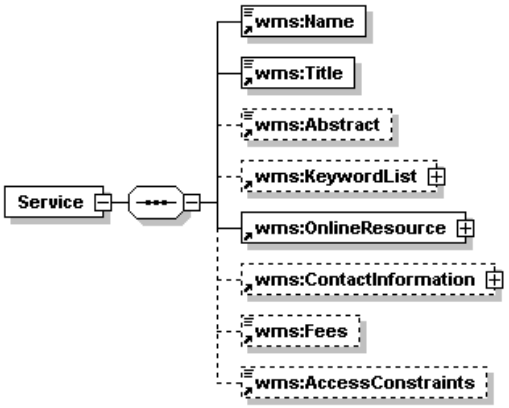


Figura 3.3: WMS – Estrutura do elemento Service

Listagem 3.1: WMS – Exemplo de meta-informação do serviço

```
8   <Service>
9     <Name>OGC:WMS</Name>
10    <Title>Tsunami_Diasaster_Data</Title>
11    <Abstract>This WMS server contains data used in http://www.mapsherpa.com
          /tsunami/. Much of the reference data is cascaded from other WMS
          servers. Before using this data in your live application, please
          consult the metadata url for each layer to check to see if you can
          legally use the data in your application.</Abstract>
12    <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="
          http://www.mapsherpa.com/cgi-bin/wms_iodra?"/>
13  </Service>
```

O elemento Capability contém meta-informação específica sobre as operações e mapas disponibilizados pelo servidor. Como se pode ver na figura 3.4, este elemento contém dois elementos obrigatórios e três elementos opcionais. Por simplicidade, serão apenas mencionados os mais relevantes:

Request contém a lista de operações suportadas pelo serviço.

Exception indica os formatos para reportar erros que são suportados.

Layer (opcional) informação sobre as camadas suportadas pelo serviço.

O elemento Layer é o mais importante, é através dele que são publicados os mapas que o serviço devolve, sendo que para cada mapa fornecido pelo serviço existe um elemento Layer correspondente.

A estrutura, os elementos e atributos que compõem um Layer são esquematizados na figura 3.5:

- Elementos:

  Name (opcional) utilizado tipicamente para interacções entre serviços.

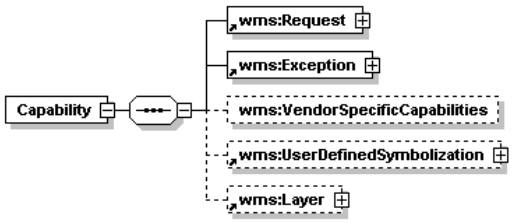


Figura 3.4: WMS – Estrutura do elemento Capability

Title descrição para ser visualizada por humanos.

Abstract (opcional) descrição em formato narrativo sobre esta camada.

KeywordList (opcional) lista de palavras-chave para que seja facilitada a pesquisa por esta camada.

SRS (opcional) sistema de coordenadas em que os dados estão armazenados.

LatLonBoundingBox (opcional) limites da camada no sistema de coordenadas EPSG:4326.

BoundingBox (opcional) para que os limites da camada sejam especificados em vários sistemas de coordenadas, cada camada pode conter zero ou mais elementos deste tipo. Não deve, no entanto, existir mais do que um elemento BoundingBox para o mesmo sistema de coordenadas.

Dimension (opcional) é base para suporte de dados multi-dimensionais, contém meta-informação sobre unidades de medida.

Extent (opcional) especifica que valores são válidos para uma dimensão.

Attribution (opcional) refere-se ao fornecedor de dados da camada.

AuthorityURL e Identifier (opcionais) um serviço pode fazer referência a identificadores definidos por entidades externas externas. Um elemento Identifier refere-se sempre a um elemento AuthorityURL – autoridade externa – e ao identificador definido externamente.

MetadataURL (opcional) meta-informação acerca dos dados que compõem a camada, ou seja, a norma a que estes dados respeitam.

DataURL (opcional) se os dados que compõem esta camada não são normalizados, então deve utilizar-se este elemento para referenciar mais informação acerca dos mesmos.

FeatureListURL (opcional) lista de entidades representadas nesta camada.

Style (opcional) nome a ser utilizado para que um estilo seja pedido.

ScaleHint (opcional) escala mínima e máxima que podem ser utilizadas para representar esta camada.

Layer (opcional) um ou mais elementos Layer que que são hierarquicamente contidos por um outro Layer.

- Atributos: todos os atributos de um Layer são opcionais e são sempre valores inteiros.

queryable pode conter os valores 0 ou 1, 1 se o servidor suporta a operação GetFeatureInfo para o elemento Layer respectivo.

cascaded pode conter um valor superior ou igual a zero. Um valor maior que zero significa que esta camada foi retransmitida de um outro servidor.

opaque se os dados da camada são maioritariamente transparentes, este valor é 0, caso contrário 1.

noSubsets se este valor é 1, significa que o serviço não tem a capacidade de devolver uma região do mapa, apenas todo o mapa, se for 0, o servidor pode devolver regiões do mapa.

fixedWidth e fixedHeigh quando estes valores são diferentes de zero, o serviço só retorna o mapa no seu tamanho e resolução originais.

Uma propriedade que é imediatamente realçada nesta estrutura é o facto de um elemento Layer poder conter outros elementos do mesmo tipo. Esta propriedade é muito importante e traduz uma característica fundamental deste elemento – **herança**.

A herança entre Layers não corresponde à herança de que se fala, por exemplo, na programação orientada por objectos, mas sim a um caso particular, onde existem regras bem definidas que especificam de que forma é que a herança afecta cada um dos elementos ou atributos de um Layer. Existem três tipos de herança:

Herança por cópia as propriedades do Layer são copiadas se não há redefinição das mesmas no Layer filho.

Herança por substituição as propriedades do Layer são substituidas se forem redefinidas no Layer filho.

Herança por adição às propriedades do Layer são adicionadas outras que são definidas no Layer filho.

Existem ainda algumas propriedades que não são herdadas. Os elementos de um Layer que suportam:

herança por substituição ou cópia são: LatLonBoundingBox, BoundingBox, Extent, Attribution, ScaleHint e os atributos do elemento Layer.

herança por adição são: Style, Dimension, AuthorityURL.

Todos os outros não suportam qualquer tipo de herança.

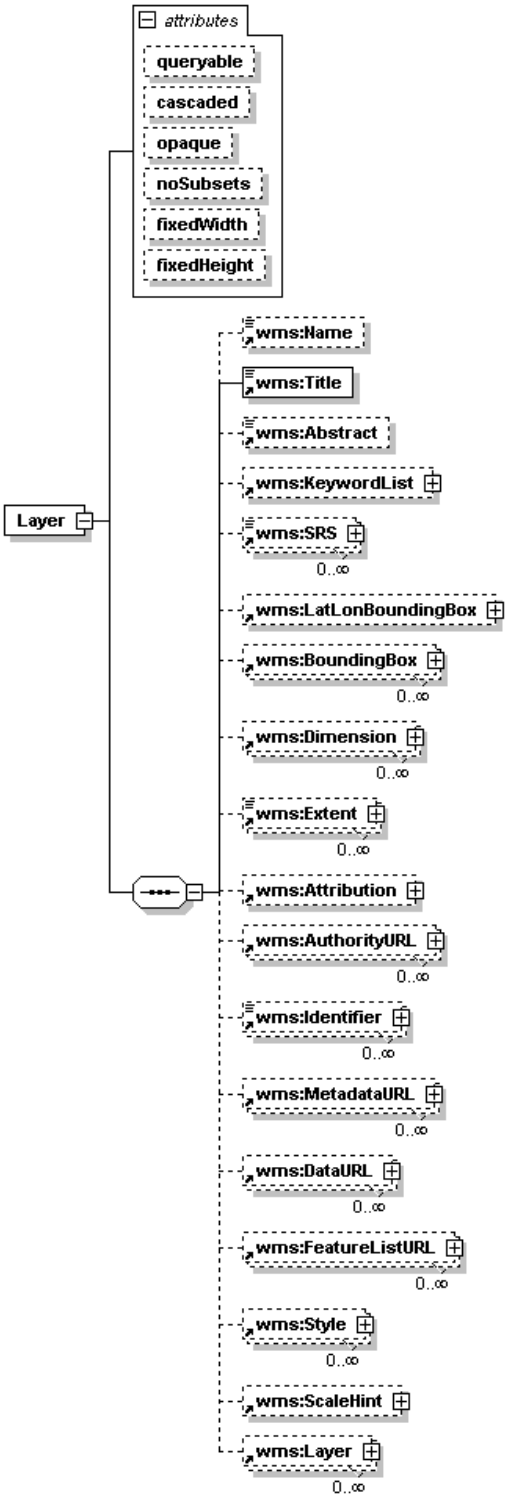


Figura 3.5: WMS – Estrutura do elemento Layer

Segue-se um extracto de um documento XML que que mostra duas camadas retornadas pela operação GetCapabilites.

Listagem 3.2: WMS – Exemplo de utilização do elemento Layer

```
104       <Layer>
105         <Name>Satellite_Imagery</Name>
106         <Title>Satellite_Imagery</Title>
107         <Abstract>Satellite_Imagery</Abstract>
108         <Layer queryable="0" opaque="0" cascaded="1">
109           <Name>GAEL_WORLD_MARCH</Name>
110           <Title>ENVISAT MERIS mosaic</Title>
111           <SRS>EPSG:4326</SRS>
112           <MetadataURL type="TC211">
113             <Format>text/html</Format>
114             <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                    xlink:type="simple" xlink:href="http://mapserv2.esrin.esa.it/
                    ionicweb/map/GAEL_WORLD_MARCH?"/>
115           </MetadataURL>
116         </Layer>
117         <Layer queryable="0" opaque="0" cascaded="0">
118           <Name>srtm_tiles</Name>
119           <Title>SRTM 90m DEM - unclassified</Title>
120           <Abstract>Hole-filled seamless SRTM data V1, 2004, International
                Centre for Tropical Agriculture (CIAT). Processed by the
                Laboratory for Applied Geomatics and GIS Science (LAGGISS),
                University of Ottawa.</Abstract>
121           <SRS>EPSG:4326</SRS>
122           <MetadataURL type="TC211">
123             <Format>text/html</Format>
124             <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                    xlink:type="simple" xlink:href="http://gisweb.ciat.cgiar.org/
                    sig/90m_data_tropics.htm"/>
125           </MetadataURL>
126         </Layer>
127         <Layer queryable="1" opaque="0" cascaded="1">
128           <Name>thai-LandsatTrue</Name>
129           <Title>Landsat Mosaic of Thailand</Title>
130           <Abstract>Landsat Mosaic of Thailand. Provided by the Asian
                Institute of Technology (AIT), Pathumthani, Thailand.</Abstract
                >
131           <SRS>epsg:4326</SRS>
132           <MetadataURL type="TC211">
133             <Format>text/html</Format>
134             <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                    xlink:type="simple" xlink:href="http://203.159.10.13/cgi-bin/
                    mapserv?map=/var/www/html/gms/thaiwms.map"/>
135           </MetadataURL>
136         </Layer>
137         <Layer queryable="0" opaque="0" cascaded="0">
138           <Name>sri_landsat</Name>
139           <Title>Landsat Mosaic of Sri Lanka</Title>
140           <Abstract>Landsat ETM+ image of Sri Lanka, obtained from https://
                zulu.ssc.nasa.gov/mrsid/mrsid.pl. Processed by the Laboratory
                for Applied Geomatics and GIS Science (LAGGISS), University of
                Ottawa.</Abstract>
```

```
141             <SRS>EPSG:4326</SRS>
142             <MetadataURL type="TC211">
143               <Format>text/html</Format>
144               <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                      xlink:type="simple" xlink:href="https://zulu.ssc.nasa.gov/
                      mrsid/docs/GeoCover_circa_2000_Product_Description.pdf"/>
145             </MetadataURL>
146           </Layer>
147         </Layer>
```

3.3.2 GetMap

A operação GetMap retorna um mapa no formato requerido pelo cliente. Se o **pedido** não puder ser completado, por exemplo porque o formato não é suportado, será retornada uma excepção.

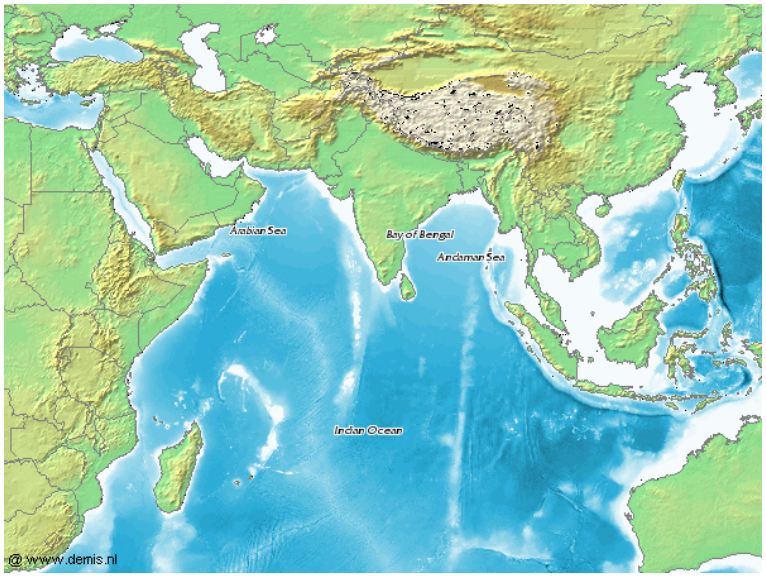


Figura 3.6: WMS – Retorno da operação GetMap

Os parâmetros principais de um pedido GetMap são os seguintes:

VERSION versão do serviço.

REQUEST neste caso o valor a ser utilizado é "GetMap".

LAYERS uma lista de camadas a ser retornada por este pedido, os valores permitidos nesta lista são os especificados no ficheiro de configuração do serviço. Esta lista segue o formato LAYERS=<layer1>,<layer2>,<layer3>, onde <layer1>, <layer2> e <layer3> correspondem ao conteúdo do elemento Name filho de Layer.

STYLES uma lista de estilos que especifica como vão ser desenhadas cada uma das camadas listadas anteriormente pelo parâmetro LAYERS. Existe uma correspondência

entre cada posição desta lista e a posição correspondente na lista especificada no parâmetro LAYERS, ou seja, a primeira camada desenhada com base no primeiro estilo listado, a segunda com base no segundo e assim sucessivamente. Existe também a possibilidade de omitir o estilo para uma, ou todas, as camadas por forma a que seja utilizado o estilo por omissão. Por exemplo, se se pretende omitir o estilo para a segunda camada utiliza-se o padrão STYLES=<estilo1>,,<estilo3>, se se pretende que seja utilizado o estilo definido por omissão para todas as camadas utiliza-se o valor STYLES=.

SRS este parâmetro indica que sistema de coordenadas vai ser utilizado para todos os valores do parâmetro seguinte – BBOX.

BBOX permite ao cliente delimitar o mapa pretendido, caso isso seja suportado pelo servidor. O formato que deve ser utilizado para este parâmetro é BBOX=<minx>,<miny>,<maxx>,<maxy>, onde minx, miny, maxx e maxy são valores reais.

WIDTH um valor inteiro que indica a largura, em píxeis, da imagem a ser retornada.

HEIGHT um valor inteiro que indica a altura, em píxeis, da imagem a ser retornada.

FORMAT (opcional) formato do mapa a ser retornado.

TRANSPARENT (opcional) indica se a cor de fundo do mapa retornado deve ser ou não transparente. Sendo um parâmetro opcional, se for omitido o WMS considera que o seu valor é FALSE.

BGCOLOR (opcional) cor de fundo a ser utilizada pelo WMS para preencher os píxeis de fundo quando o parâmetro TRANSPARENT tem o valor FALSE.

Um exemplo de um pedido a um WMS é dado pelo URL:
http://www.mapsherpa.com/cgi-bin/wms_iodra?SERVICE=wms&REQUEST=getMap
&LAYERS=Bathymetry,Topography,Hillshading,borders,coastlines,oceans
&format=image/png&bbox=21.0123,-34.4092,133.253,49.6596&STYLES=
&VERSION=1.1.1&SRS=EPSG:4326.

Nesta invocação é pedido um mapa no formato image/png com 300 píxeis de altura e 400 de largura. A **resposta** a este pedido inclui as camadas: Bathymetry, Topography, Hillshading, borders e coastlines, todas elas com base no estilo definido por omissão. O retorno a este pedido pode ser visto na figura 3.6.

3.3.3 GetFeatureInfo

Esta operação é opcional e permite a um cliente requerer mais informação acerca de um mapa retornado pela operação GetMap. Tipicamente, é utilizado no caso em que um cliente, após invocar a operação GetMap, pretende obter mais informação acerca de um ponto – à sua escolha – do mapa. Assim sendo, o cliente, no pedido GetFeatureInfo indica sobre que pixel – através das suas coordenadas – quer mais informação e em que camadas deve essa

informação ser obtida. Um detalhe muito importante acerca de um pedido GetFeatureInfo prende-se com o facto do WMS não ter a noção de estado, pelo que, o cliente, para além de ter que indicar no pedido os parâmetros referentes à operação GetFeatureInfo em particular, tem também que indicar os parâmetros da operação GetMap (todos os mencionados na secção 3.3.2 com a excepção dos parâmetros VERSION e REQUEST) cujo retorno se pretende que seja inquirido.

Os parâmetros principais de um **pedido** GetFeatureInfo são então os seguintes:

VERSION versão do serviço a utilizar.

REQUEST neste caso o valor a ser utilizado é "GetFeatureInfo"

map_request_copy parâmetros da operação GetMap cujo retorno se pretende que seja inquirido.

QUERY_LAYERS lista das camadas serão inquiridas no formato LAYERS=<layer1>,<layer2>,<layer3>, onde <layer1>, <layer2> e <layer3> correspondem ao conteúdo do elemento Name filho de Layer.

INFO_FORMAT (opcional) o formato que deve ser utilizado para o retorno da operação. De salientar que os valores permitidos neste parâmetro foram previamente declarados no ficheiro de configuração do serviço no elemento <Format> filho de <FeatureInfo> que por sua vez é filho de <Request>.

X coordenada, em píxeis, no eixo dos xx do ponto sobre o qual a operação irá inquirir.

Y coordenada, em píxeis, no eixo dos yy do ponto sobre o qual a operação irá inquirir.

Um exemplo de um pedido deste tipo é o URL seguinte:
http://www.mapsherpa.com/cgi-bin/wms_iodra?SERVICE=wms
&REQUEST=getFeatureInfo&QUERY_LAYERS=locations&X=315&Y=231
&LAYERS=Bathymetry,Topography,Hillshading,borders,coastlines,oceans,
locations&format=image/png&bbox=21.0123,-34.4092,133.253,49.6596&STYLES=
&VERSION=1.1.1&SRS=EPSG:4326&INFO_FORMAT=text/plain.

O retorno deste pedido é mostrado na listagem seguinte:

Listagem 3.3: WMS – Exemplo de retorno de uma invocação a uma operação GetFeatureInfo

```
1 <ll:FeatureCollection xmlns:gml="http://www.opengis.net/gml" xmlns:ll="http:
      //www.lat-lon.de" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
2   <gml:boundedBy>
3     <gml:Box srsName="EPSG:4326">
4       <gml:coordinates>7.93,52.22 8.19,52.34</gml:coordinates>
5     </gml:Box>
6   </gml:boundedBy>
7 </ll:FeatureCollection>
```

3.4 Web Feature Service

Este serviço foi concebido pelo consórcio OpenGIS para retornar informação geográfica discreta, pronta a ser manipulada. A informação retornada é codificada em *Geographic Markup Language* (GML), sendo o GML uma recomendação do mesmo consórcio.

Este serviço suporta operações de inserção, actualização, remoção, inquérito e descoberta de entidades geográficas. As operações definidas para este serviço são (ver figura 3.7):

GetCapabilities que retorna meta-informação do serviço, descrevendo-o e enumerando os parâmetros que este aceita e operações que este suporta – secção 3.4.1.

DescribeFeatureType que permite obter uma descrição da estrutura de cada uma das entidades disponibilizadas – secção 3.4.1.

GetFeature retorna a IG pretendida em GML para uma entidade especificada pelo cliente – secção 3.4.3.

Transaction (opcional) trata-se de um pedido que visa modificar o conteúdo do repositório de IG mantido por um dado WFS – secção 3.4.4.

LockFeature que permite bloquear o acesso a uma determinada entidade durante uma transacção.

De acordo com (OpenGIS, 2002a), existem dois tipos de WFS. Esta tipificação dos WFS depende das operações que estes suportam: um *WFS básico* suporta as operações GetCapabilities, DescribeFeatureType e GetFeature, por outro lado, um *WFS transaccional* suporta, para além das operações suportadas pelo *WFS básico*, também a operação Transaction e opcionalmente a operação LockFeature.

Para este serviço, os pedidos exemplificados estão codificados em XML, e foram enviados via HTTP POST.

3.4.1 GetCapabilities

Um **pedido** GetCapabilities codificado em XML, na versão 1.0.0, tem que estar conforme o XML Schema respectivo[4], apresentado na listagem seguinte.

[4] `http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd`

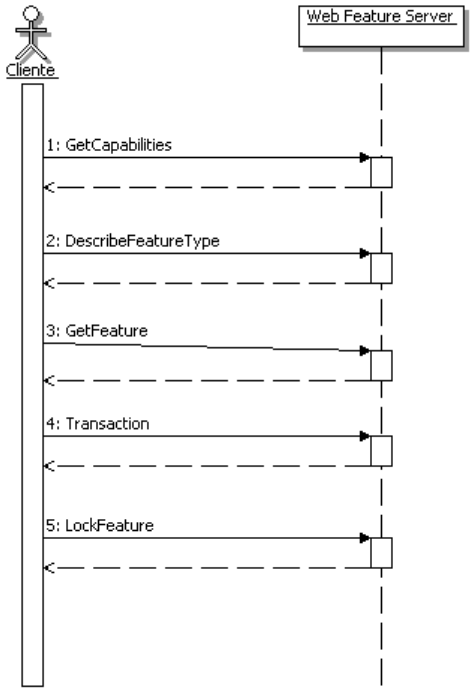


Figura 3.7: WFS – Operações disponibilizadas

Listagem 3.4: WFS – Excerto do XML Schema que define a estrutura de um pedido GetCapabilities

```
51  <xsd:element name="GetCapabilities" type="wfs:GetCapabilitiesType"/>
52  <xsd:complexType name="GetCapabilitiesType">
53    <xsd:attribute name="version" type="xsd:string" use="optional" fixed="
          1.0.0"/>
54    <xsd:attribute name="service" type="xsd:string" use="required" fixed="
          WFS"/>
55  </xsd:complexType>
```

Da listagem anterior, verifica-se que um pedido GetCapabilities tem apenas um elemento obrigatório – GetCapabilities – que pode ter dois atributos – version e service, o primeiro especifica a versão do serviço que se pretende invocar, o segundo o serviço que se pretende invocar. Note-se que o atributo service para o WFS tem um valor fixo e obrigatório: "WFS".

A listagem seguinte é um exemplo de um pedido GetCapabilities a um WFS.

Listagem 3.5: WFS – Exemplo de um pedido GetCapabilities

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <GetCapabilities service="WFS"/>
```

A **resposta** da invocação da operação GetCapabilities também é formalmente especificada por um XML Schema [5] que será detalhado seguidamente.

A raiz do documento retornado pela operação GetCapabilities (ver figura 3.8) é composta por quatro elementos – Service, Capability, FeatureTypeList e Filter_Capabilities.

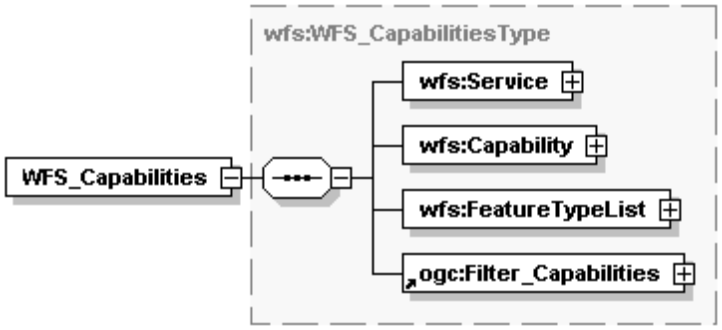


Figura 3.8: WFS – Estrutura global de uma resposta GetCapabilities

O elemento Service (ver figura 3.9) contém meta-informação do serviço, permitindo que este se auto-descreva é composto pelos seguintes elementos:

Name nome que o fornecedor do serviço atribui à instância do mesmo.

Title uma descrição curta do serviço, para ser utilizada para efeitos de listagem apresentada a um utilizador.

Abstract (opcional) trata-se também de uma descrição, um pouco mais extensa que a anterior, e num formato narrativo.

Keywords (opcional) lista de palavras-chave que permitam ao serviço ser catalogado e pesquisado através dessas palavras-chave.

OnlineResource pode ser utilizado, por exemplo, para referenciar o sítio do fornecedor deste serviço.

Fees (opcional) se o serviço tiver algum tipo de taxa associada à sua utilização, essa informação deve ser incluída aqui. No entanto, no caso de não existir qualquer taxa, deve utilizar-se a palavra reservada none (<Fees>none</Fees>).

Access Constraints (opcional) no caso do serviço estar sujeito a restrições de acesso ou utilização, pode usar-se este elemento. De salientar que não existe qualquer tipo de sintaxe definida para este efeito. No caso de não existir qualquer restrição, deve utilizar-se a palavra reservada none (<AccessConstraints>none</AccessConstraints>).

Um exemplo da utilização do elemento Service é apresentado de seguida:

[5] `http://schemas.opengis.net/wfs/1.0.0/WFS-capabilities.xsd`

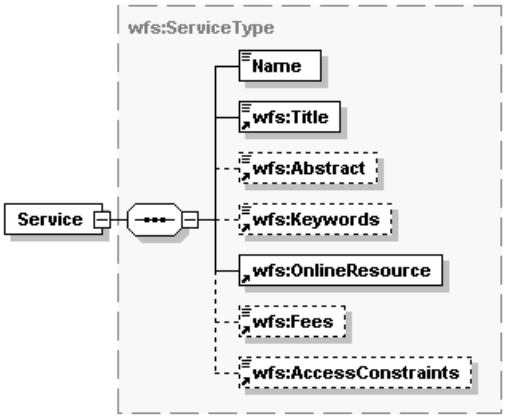


Figura 3.9: WFS – Elemento Service

Listagem 3.6: WFS – Exemplo da utilização do elemento Service no retorno da operação GetCapabilities

```
2  <Service>
3    <Name>WebFeatureServer</Name>
4    <Title>deegree WFS</Title>
5    <Abstract>Web Feature Server para testes</Abstract>
6    <OnlineResource>http://127.0.0.1:8080/deegreewfs</OnlineResource>
7  </Service>
```

O elemento Capability contém meta-informação acerca dos pedidos que o WFS suporta. Este elemento é composto por outros dois elementos (ver figura 3.10) – Request e VendorSpecificCapabilities (opcional) – o segundo permite a um fornecedor listar as operações proprietárias do serviço, o primeiro é composto por um conjunto de elementos que detalham as operações que o serviço suporta:

GetCapabilities especifica os protocolos que podem ser utilizados para a invocação da operação GetCapabilities. Actualmente somente HTTP POST e/ou GET podem ser utilizados.

DescribeFeatureType especifica os protocolos que podem ser utilizados para a invocação da operação DescribeFeatureType. Adicionalmente, permite identificar que XML Schemas são suportados pelo serviço, sendo que é obrigatório que este suporte XML Schema.

Transaction especifica os protocolos que podem ser utilizados para a invocação da operação Transaction.

GetFeature especifica os protocolos que podem ser utilizados para a invocação da operação GetFeature. Adicionalmente, identifica os formatos de retorno suportados pelo serviço, sendo que, o GML tem que ser suportado obrigatoriamente.

GetFeatureWithLock especifica os protocolos que podem ser utilizados para a invocação da operação GetFeatureWithLock.

LockFeature especifica os protocolos que podem ser utilizados para a invocação da operação LockFeature.

Para concluir a apresentação do elemento Capability, apresenta-se um exemplo da utilização do mesmo:

Listagem 3.7: WFS – Exemplo da utilização do elemento Capability do retorno da operação GetCapabilities

```
8  <Capability>
9    <Request>
10     <GetCapabilities>
11       <DCPType>
12         <HTTP>
13           <Get onlineResource="http://127.0.0.1:8080/deegreewfs/wfs"/>
14           <Post onlineResource="http://127.0.0.1:8080/deegreewfs/wfs"/>
15         </HTTP>
16       </DCPType>
17     </GetCapabilities>
18     <DescribeFeatureType>
19       <SchemaDescriptionLanguage>
20         <XMLSCHEMA/>
21       </SchemaDescriptionLanguage>
22       <DCPType>
23         <HTTP>
24           <Get onlineResource="http://127.0.0.1:8080/deegreewfs/wfs"/>
25           <Post onlineResource="http://127.0.0.1:8080/deegreewfs/wfs"/>
26         </HTTP>
27       </DCPType>
28     </DescribeFeatureType>
29     <GetFeature>
30       <ResultFormat>
31         <XML/>
32         <GML2/>
33         <FEATURECOLLECTION/>
34       </ResultFormat>
35       <DCPType>
36         <HTTP>
37           <Post onlineResource="http://127.0.0.1:8080/deegreewfs/wfs"/>
38         </HTTP>
39       </DCPType>
40     </GetFeature>
41   </Request>
42 </Capability>
```

O elemento FeatureTypeList permite listar as entidades que o serviço disponibiliza – por cada entidade suportada existe um elemento FeatureType que a descreve – associando a cada uma destas as operações de inquérito, actualização e transacção que podem ser efectuadas. O suporte destas operações é dado pela inclusão dos respectivos elementos dentro do elemento Operations. Este elemento ocorre dentro do elemento FeatureTypeList (ver figura 3.11) – como forma de definir globalmente, ou seja, para todas as entidades, que operações são suportadas – ou dentro do elemento FeatureType (ver figura 3.11) – como forma de definir as operações suportadas por uma entidade em particular:

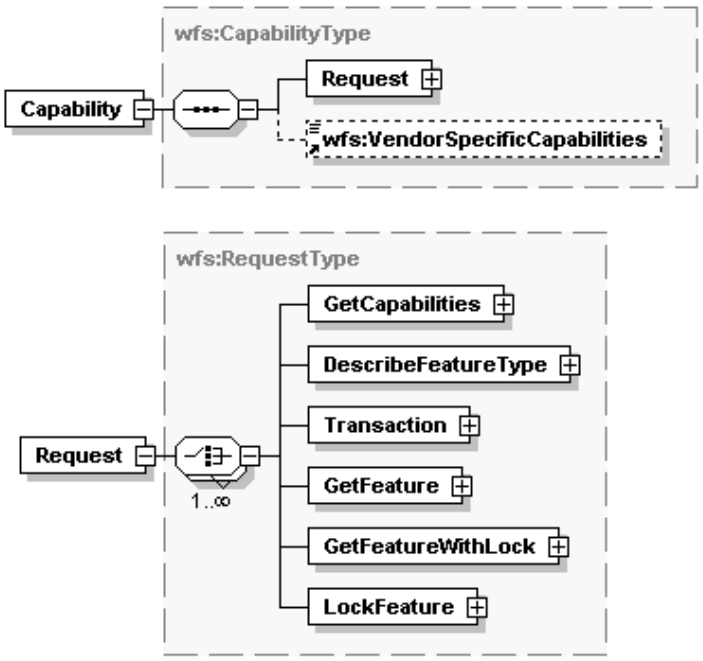


Figura 3.10: WFS – Elemento capability

Insert utilizado para indicar que o serviço permite criar uma instância de uma entidade.

Update utilizado para indicar que o serviço permite actualizar uma instância de uma entidade.

Delete utilizado para indicar que o serviço permite eliminar uma instância de uma entidade.

Query utilizado para indicar que o serviço permite inquirir sobre as instâncias de uma entidade.

Lock utilizado para indicar que o serviço permite bloquear instâncias de uma entidade.

O exemplo seguinte define três entidades: Europe, Cities e Rivers, todas elas permitindo operações de inserção e inquérito.

Listagem 3.8: WFS – Exemplo da utilização do elemento FeatureTypeList no retorno da operação GetCapabilities

```
43   <FeatureTypeList>
44     <Operations>
45       <Query/>
46       <Insert/>
47     </Operations>
48     <FeatureType>
```

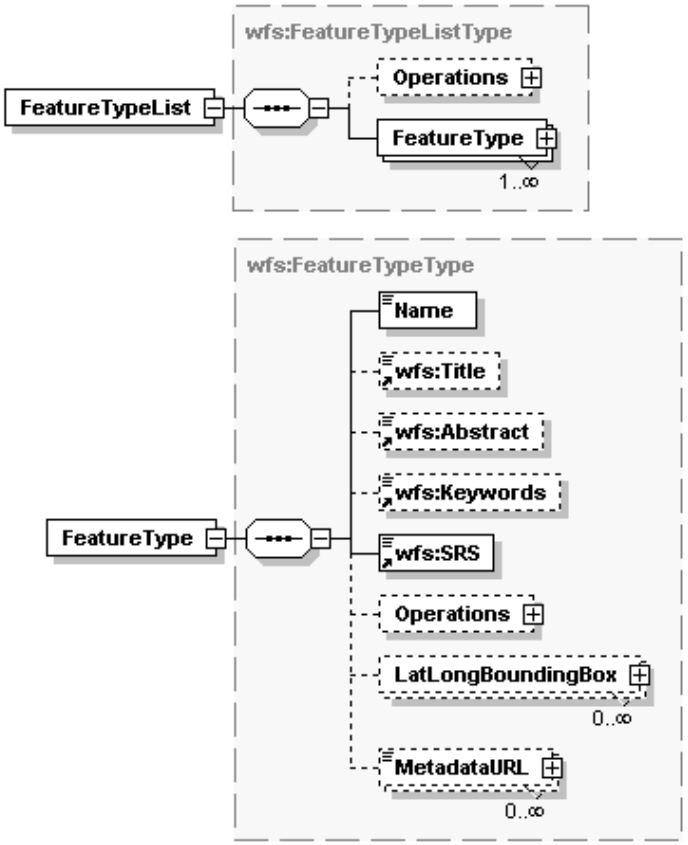


Figura 3.11: WFS – O elemento FeatureTypeList

```
49        <Name>Europe</Name>
50        <Title>european bordes</Title>
51        <Keywords/>
52        <SRS>EPSG:4326</SRS>
53        <LatLonBoundingBox minx="-32.0" miny="28.0" maxx="44.0" maxy="70.0"/>
54        <Operations>
55          <Query/>
56          <Insert/>
57        </Operations>
58      </FeatureType>
59      <FeatureType>
60        <Name>Cities</Name>
61        <Title>mayor cities of central europe</Title>
62        <Keywords/>
63        <SRS>EPSG:4326</SRS>
64        <LatLonBoundingBox minx="3.7" miny="46.0" maxx="17.1" maxy="56.2"/>
65        <Operations>
66          <Query/>
67          <Insert/>
68        </Operations>
69      </FeatureType>
70      <FeatureType>
71        <Name>Rivers</Name>
72        <Keywords/>
73        <SRS>EPSG:4326</SRS>
74        <LatLonBoundingBox minx="-21.0" miny="36.0" maxx="62.0" maxy="70.0"/>
75        <Operations>
76          <Query/>
77          <Insert/>
78        </Operations>
79      </FeatureType>
80    </FeatureTypeList>
```

3.4.2 DescribeFeatureType

Esta operação permite pedir ao WFS que retorne um XML Schema relativo a uma dada entidade, informando assim o cliente acerca de como está codificada a informação dessa entidade.

O exemplo de um **pedido** DescribeFeatureType é apresentado na listagem seguinte:

Listagem 3.9: WFS – Exemplo de um pedido DescribeFeatureType

```
<?xml version="1.0"?>
<DescribeFeatureType version="1.0.0" service="WFS"
      xmlns="http://www.opengis.net/wfs"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <TypeName>Europe</TypeName>

</DescribeFeatureType>
```

Através deste pedido, o cliente está a solicitar informação acerca da estrutura da entidade Europe. A **resposta** a esta operação é um XML Schema cuja estrutura aparece ilustrada na figura 3.12.

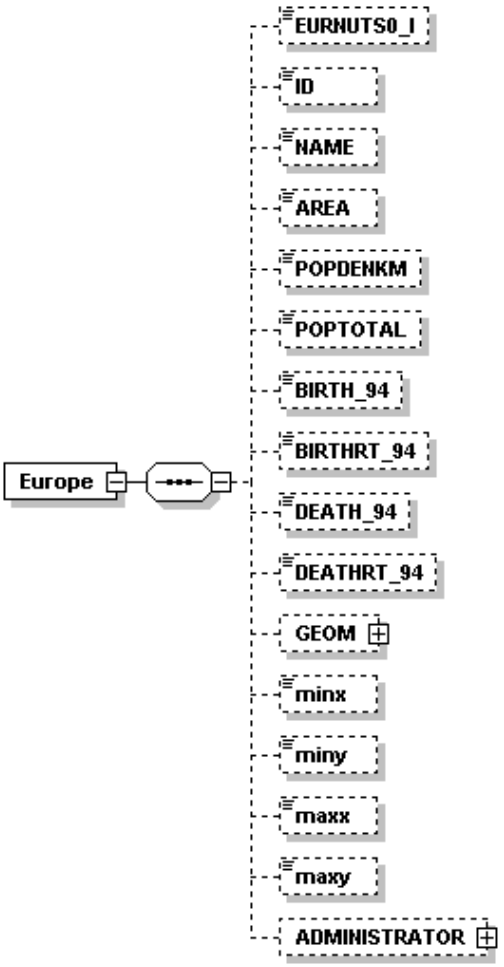


Figura 3.12: WFS – Estrutura de um XML Schema retornado por um pedido DescribeFeatureType

O retorno de um XML Schema descrevendo a estrutura a entidade *Europe*, vai permitir a um cliente processar a informação de acordo com essa mesma estrutura, ou seja, num pedido deste tipo, o foco é na sintaxe da informação e não na informação per si.

3.4.3 GetFeature

A operação GetFeature permite inquirir o WFS no sentido de obter as entidades do mesmo. Ambos o pedido e resposta são codificados em XML.

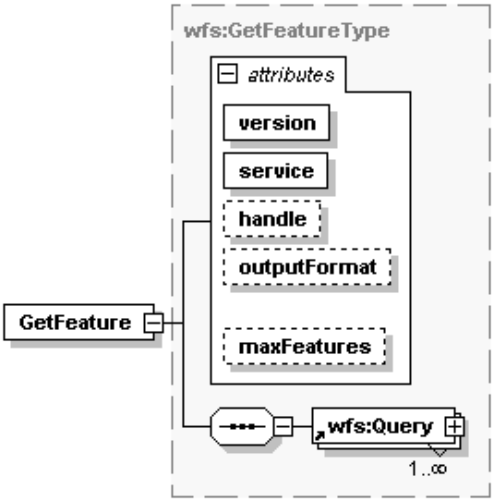


Figura 3.13: WFS – O Elemento GetFeature

Um **pedido** deste tipo é codificado de acordo com o ilustrado na figura 3.13. O pedido pode conter cinco atributos:

version versão do serviço a utilizar. Este valor é fixo para cada versão do WFS.

service o serviço a inquirir. Para o caso do WFS este valor é "WFS".

handle (opcional) permite que a aplicação cliente associe um nome ao pedido. Este nome tem significado somente para o cliente, e o servidor utiliza-o em caso de erro como forma de identificação do elemento afectado.

outputFormat (opcional) o formato em que vai ser codificado o resultado. Por omissão está definido o GML.

maxfeatures (opcional) o número máximo de entidades que o pedido pode retornar.

Adicionalmente, contidos no elemento GetFeature estão um ou mais elementos Query, que representam os inquéritos que o cliente faz ao serviço. Este elemento define os tipos das entidades a inquirir, as propriedades requeridas e as restrições a aplicar.

A listagem seguinte apresenta um pedido cujo objectivo é obter os contornos da Europa, isto é feito apenas através de um elemento Query:

Listagem 3.10: WFS – Pedido GetFeature cujo objectivo é obter os contornos da Europa

```
<?xml version="1.0" encoding="iso-8859-1"?>
<wfs:GetFeature outputFormat="GML2" xmlns:gml="http://www.opengis.net/gml"
    xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/
    ogc">
  <wfs:Query typeName="Europe">
    <ogc:Filter>
      <ogc:BBOX>
        <ogc:PropertyName>/Europe/Border</ogc:PropertyName>
        <gml:Box>
          <gml:coord>
            <gml:X>1</gml:X>
            <gml:Y>40</gml:Y>
          </gml:coord>
          <gml:coord>
            <gml:X>12</gml:X>
            <gml:Y>56</gml:Y>
          </gml:coord>
        </gml:Box>
      </ogc:BBOX>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>
```

A **resposta** a um pedido GetFeature é um documento XML no formato especificado pelo atributo outputFormat mencionado anteriormente. No pedido exemplificado anteriormente o formato especificado foi "GML2", pelo que a resposta do mesmo é um documento GML com a informação solicitada.

Uma ferramenta simples para efectuar alguns testes a Geo WS é o *deegree Viewer/Converter*[6], fornecida no âmbito do projecto deegree. Para demonstrar a resposta ao pedido mostrado anteriormente recorreu-se a esta aplicação. O resultado pode ser visto na figura 3.14, onde são mostrados os contornos da Europa visualmente e não em GML, conforme foram recebidos.

3.4.4 Transaction

Esta operação tem como objectivo a execução de operações de actualização sobre uma ou mais entidades de um serviço. Ao ser opcional, um WFS não necessita de a implementar para estar conforme a norma do OpenGIS.

Um **pedido** deste tipo é codificado em XML e tem a estrutura ilustrada na figura 3.15, estando todo o pedido dentro do elemento Transaction.

Este elemento contém até quatro atributos:

[6] `http://deegree.sourceforge.net/src/demos.html#viewer`

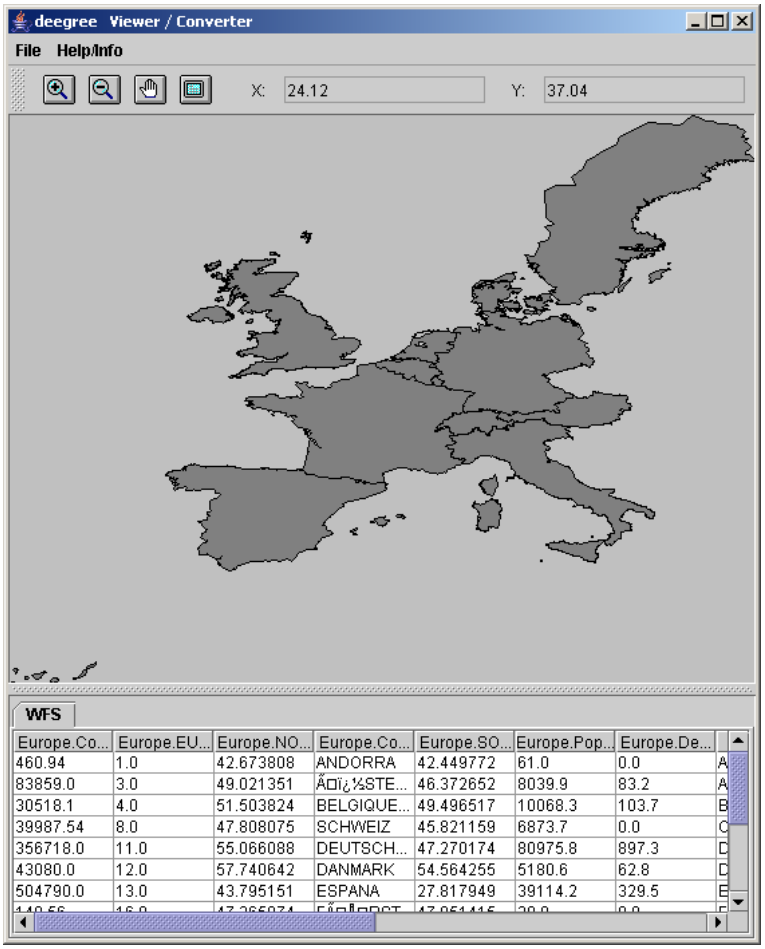


Figura 3.14: WFS – A resposta a um pedido GetFeature

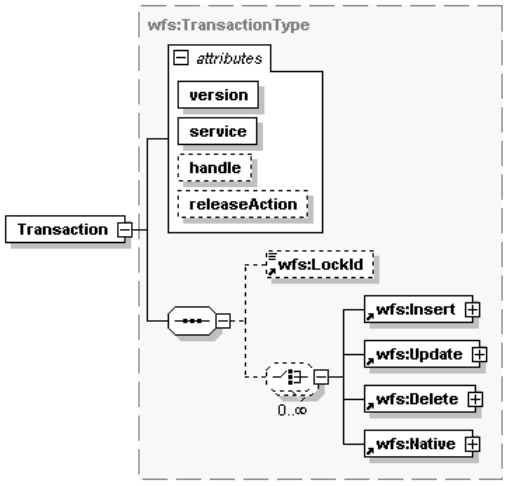


Figura 3.15: WFS – A estrutura de um pedido Transaction

version versão do serviço a utilizar. Este valor é fixo para cada versão do WFS.

service o serviço a inquirir. Para o caso do WFS este valor é "WFS".

handle (opcional) permite que a aplicação cliente associe um nome ao pedido. Este nome tem significado somente para o cliente, e o servidor utiliza-o em caso de erro como forma de identificação do elemento afectado.

releaseAction (opcional) este atributo só tem significado se o serviço suporta as operações LockFeature ou GetFeatureWithLock e define de que forma serão tratadas as entidades bloqueadas no final da transacção.

Adicionalmente o elemento Transaction pode conter um elemento LockId que indica que a operação vai ser feita sobre um conjunto de entidades previamente bloqueadas. Pode conter também n elementos Insert, Delete e Update, para inserção, remoção e actualização, respectivamente, de entidades. Existe também a possibilidade de utilizar o elemento Native que serve para referenciar operações ou comandos proprietários do fornecedor do serviço e que, por consequência, não obedecem à norma do OpenGIS.

A listagem seguinte é o exemplo de um pedido desta operação que, como se pode ver, insere uma nova cidade com o *ID 2224.0* na base de dados.

Listagem 3.11: WFS: Exemplo de um pedido Transaction

```
<?xml version="1.0" encoding="UTF-8"?>
<Transaction xmlns:gml="http://www.opengis.net/gml">
  <Insert>
    <Cities fid="ID2">
      <Cities.ID>2224.0</Cities.ID>
      <Cities.Status>Provincial capital</Cities.Status>
      <Cities.PopulationClass>100,000 to 250,000</Cities.PopulationClass>
      <Cities.AdminArea>NRW</Cities.AdminArea>
      <Cities.Name>Euskirchen</Cities.Name>
      <Cities.location>
        <gml:Point srsName="EPSG:4326">
          <gml:coordinates cs="," decimal="." ts="␣">6.79,50.66</
              gml:coordinates>
        </gml:Point>
      </Cities.location>
    </Cities>
  </Insert>
</Transaction>
```

A **resposta** a um pedido deste tipo é codificada em XML e contém o resultado da operação. A estrutura desse documento XML é ilustrada na figura 3.16 e é composta pelos seguintes elementos:

InsertResult (opcional) este elemento só aparece quando o pedido correspondente contém o elemento Insert para indicar o identificador(es) da(s) entidade(s) inserida(s). Existe um elemento destes para cada elemento Insert presente no pedido.

TransactionResult serve para declarar o estado da operação, através da existência do elemento Status, que pode conter um de três elementos representativos do estado a transacção:

SUCCESS significa que a operação foi concluída com sucesso.

FAILED pelo menos uma excepção ocorreu no processamento da operação.

PARTIAL a operação foi parcialmente concluída, o que significa que a informação pode estar inconsistente.

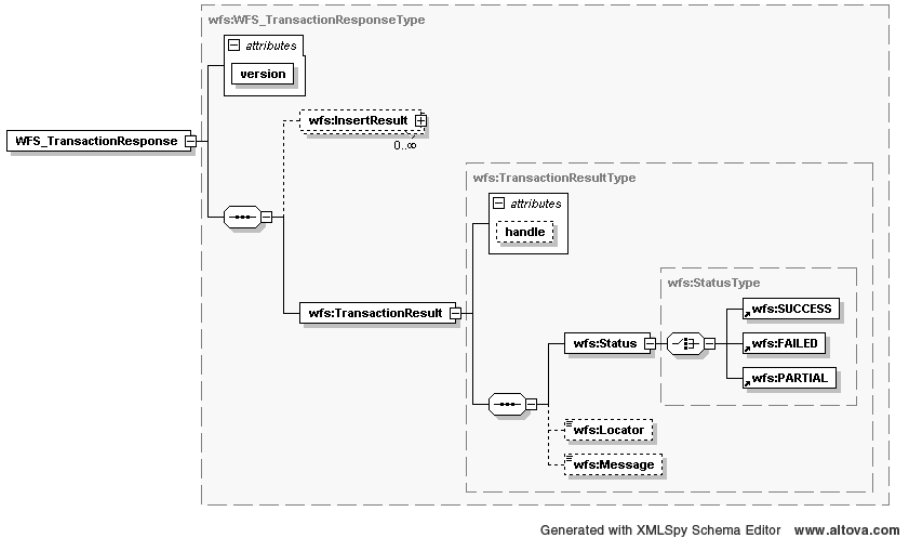


Figura 3.16: WFS – A estrutura de uma resposta Transaction

Como exemplo, apresenta-se a resposta retornada pelo servidor ao pedido mostrado anteriormente:

Listagem 3.12: WFS: Exemplo de uma resposta a um pedido Transaction

```
<WFS_TransactionResponse xmlns="http://www.opengis.net/wfs" version="1.0.0">
  <InsertResult/>
  <TransactionResult>
    <SUCCESS/>
  </TransactionResult>
</WFS_TransactionResponse>
```

Como se pode ver temos o elemento InsertResult correspondente ao elemento Insert utilizado no pedido. De seguida aparece o elemento TransactionResult que indica que a operação foi bem sucedida.

Conclui-se assim a apresentação do Geo WS preconizados pelo consórcio OpenGIS.

Quer este capítulo, quer o anterior, servem para apresentar a tecnologia de base que foi explorada neste trabalho de mestrado. Antes de discutir as limitações na descoberta e exploração automática destes serviços (capítulo 5) faz-se uma apresentação dos conceitos relacionados com a semântica, no capítulo seguinte.

Capítulo 4

Conhecimento

O estudo da tecnologia relacionada com os XML WS de uma forma geral, capítulo 2, e concretamente no âmbito da IG com os Geo WS, capítulo 3, conclui a apresentação do estado da arte na esfera de acção dos WS. Estas apresentações foram sucintas, pois não são o fim em si mesmo deste trabalho. Mas é sobre esta tecnologia dos Geo WS que se irá propor a utilização de ontologias (capítulo 6). Por isso, neste capítulo, entra-se numa vertente menos tecnológica onde se pretendem introduzir os conceitos e relações entre vocabulário, dicionário, taxinomia e thesaurus. Por fim, introduz-se o conceito de ontologia apresentando a sua definição e classificação de acordo com (Guarino, 1998). Complementarmente apresenta-se de uma forma geral a *Web Ontology Language* (OWL).

4.1 Introdução

Actualmente estes conceitos têm sido objecto de muitos estudos e aplicações no mundo das tecnologias da informação, nomeadamente quando se pretende dar aos sistemas a capacidade de fazer inferências complexas, correlacionando assuntos que são aparentemente disjuntos (Librelotto, Ramalho, and Henriques, 2003).

Em (de Almeida, 1997), Almeida estudou a problemática da especificação formal de dicionários. Saias abordou em (Saias, 2003) a extracção de informação semântica de documentos e a sua representação sob a forma de ontologias. Librelotto, em 2003, apresentou a linguagem XSTM (*XML Specification for Topic Maps*) que descreve o processo de extracção de conhecimento de documentos XML (Librelotto, Ramalho, and Henriques, 2003). Esta extracção de conhecimento culmina na criação duma ontologia.

Uma outra iniciativa de relevo neste domínio é a DCMI (*Dublin Core Metadata Initiative*) [1], organização que promove a adopção de normas para meta-informação e desenvolve vocabulários especializados para aumentar a inteligência de sistemas de descoberta de informação.

Dos trabalhos especialmente relacionados com a IG, destaca-se o TGN (*Getty Thesau-*

[1] `http://dublincore.org`

rus of Geographic Names Online [2]), desenvolvido pelo Getty (*Getty Research Institute*). É um thesaurus para IG disponível na Internet.

Um bom exemplo da falta de conhecimento semântico é o desempenho ineficiente dos motores de busca, que se baseiam na extracção de palavras-chave dos documentos que são pesquisados para comporem os seus resultados. Não têm a capacidade de analisar o significado do pedido para orientarem os resultados retornados a esse significado.

4.2 Vocabulário

Um vocabulário é uma lista alfabética de vocábulos de uma língua, ou seja, um conjunto de palavras.

Lista dos vocábulos de uma língua, em geral desacompanhados da respectiva definição ou com uma explicação muito sucinta; léxico; dicionário; terminologia em uso para dada ciência; arte, indústria, etc.[3]

Se o significado dessas palavras for comum dentro das entidades que utilizam um determinado vocabulário, não há lugar a ambiguidades e ter-se-á um vocabulário controlado, existindo portanto, uma correspondência unívoca entre o conceito e o vocábulo.

4.3 Dicionário

Um dicionário estende o vocabulário na medida em que não é apenas uma enumeração de termos. Contém também o significado desses termos. Um exemplo de uma entrada de dicionário pode ser vista na figura 4.1. As principais características de um dicionário são (Librelotto, 2005):

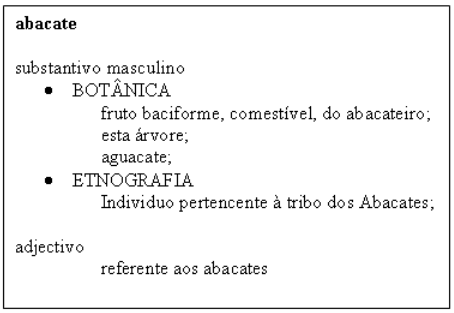
Simplicidade na medida em que pretende definir termos de forma sucinta e objectiva, de forma a eliminar ambiguidades de significado, reduzindo assim o risco de interpretações erróneas.

Adaptabilidade é constantemente actualizado à medida que vão surgindo novos termos e/ou interpretações dos já existentes.

A funcionalidade básica de dicionário digital é composta pelo conjunto de operações que se podem realizar sobre o dicionário. Entre essas operações salientam-se a consulta (de Almeida, 1997), que se resume ao fornecimento de informação sobre um termo:

$$consulta : termo \times dicionario \longrightarrow inf$$
$$consulta(t, d) \stackrel{def}{=} d(t)$$

[2] `http://www.getty.edu/research/conducting_research/vocabularies/tgn`
[3] Definição obtida no Dicionário Universal da Língua Portuguesa.



abacate

substantivo masculino
- BOTÂNICA
  fruto baciforme, comestível, do abacateiro;
  esta árvore;
  aguacate;
- ETNOGRAFIA
  Individuo pertencente à tribo dos Abacates;

adjectivo
  referente aos abacates

Figura 4.1: Exemplo de entrada num dicionário

Generalizando, um dicionário armazena a correspondência entre termos e a respectiva informação, sendo que a consulta consiste em achar a imagem correspondente a um termo. Em (de Almeida, 1997) o dicionário é formalmente modelado como:

$$dicionario = termo \rightharpoonup inf$$

$$f : termo \times dicionario \longrightarrow inf$$
$$f(t, d) \stackrel{def}{=} d(t)$$

4.4 Taxinomia

Taxinomia é a ciência que estuda a organização dos seres vivos com base nas suas semelhanças ou diferenças, ou seja, observando as características dos seres vivos faz-se uma distribuição em classes que constitui uma classificação. Esta classificação resulta precisamente da comparação dessas semelhanças e diferenças. No âmbito da classificação dos seres vivos destaca-se a contribuição de Lineu (1707-1778).

Classificação científica; nomenclatura das classificações.[4]

No entanto, este conceito, pode aplicar-se na generalidade, ou seja, uma taxinomia é uma lista estruturada em árvore, que forma uma hierarquia de vocábulos e/ou conceitos. Essa hierarquia tem como raiz o mais abstracto desses conceitos, sendo que, à medida que se desce nessa sua estrutura, o nível de abstracção diminui para que os conceitos se tornem mais específicos. Adicionalmente, há que referir que os elementos presentes nesta hierarquia herdam as características dos seus pais, avós, e assim sucessivamente.

[4]Definição obtida no Dicionário Universal da Língua Portuguesa.

O fundamental numa classificação é não olhar às características acessórias, mas sim aquelas que reflectem a essência e carácter distintivo do que está a ser classificado.

Idealmente, cada elemento da árvore deve ser único e não deve introduzir qualquer tipo de ambiguidade, mantendo uma exclusividade mútua ao longo de toda a hierarquia. Por exemplo, se "caminho" aparece como filho de "via", não deverá aparecer também como filho de "caminho-de-ferro".

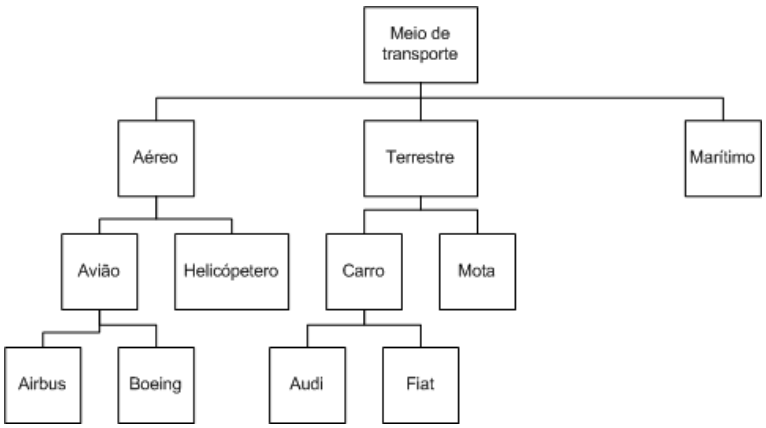


Figura 4.2: Taxinomia para Meio de Transporte

Na figura 4.2 apresenta-se uma taxinomia para "Meio de Transporte". Através desta taxinomia pode dizer-se, por exemplo, que "Audi" é um carro, que por sua vez é um "Meio de Transporte" "Terrestre".

4.5 Thesaurus

De certa forma, o thesaurus é uma taxinomia estendida por forma a que sejam incluídas e apresentadas as relações horizontais entre os elementos que a compõem.

A norma ISO 2788 (ISO, 1986) define thesaurus da seguinte forma:

> *The vocabulary of a controlled indexing language, formally organized so that a priori relationships between concepts (for example as "broader"and "narrower") are made explicit.*

Como se pode ler na definição acima, um thesaurus, organiza formalmente uma linguagem de forma a apresentar explicitamente as relações entre os diversos conceitos presentes.

Em (ISO, 1986) recomenda-se a descrição de um conceito através das seguintes propriedades:

Termo Genérico refere-se ao termo superior na hierarquia, tendo um significado mais abrangente ou menos específico. Podem existir vários termos genéricos associados a um termo específico. A propriedade inversa a esta é Termo Específico.

Termo Específico refere-se ao termo imediatamente abaixo deste na hierarquia, tendo um significado mais estrito.

Raiz ou Termo Superior é o ascendente mais alto na hierarquia de termos, isto é, percorrendo de forma recursiva os Termos Genéricos chega-se a um ponto em que o Termo Genérico não tem pai, esse é o Termo Superior.

Termo Relacionado refere-se a um termo que não é nem genérico, nem específico e que se relaciona implicitamente com o termo.

Uso referencia um termo que pode ser usado em lugar do termo actual, ou seja, um sinónimo.

Nota de contexto trata-se de uma frase associada ao termo que explica o significado do mesmo no âmbito do thesaurus, podendo ser útil em casos onde o significado não é óbvio sem a existência de uma contextualização.

Na secção 4.4 apresentou-se a taxinomia como sendo uma relação hierárquica que funcionaria para um contexto bem definido e sobre um vocabulário controlado. O thesaurus, por sua vez, não só contém as relações presentes numa taxinomia (dadas pelos termos genérico e específico), mas também contém relações semânticas entre os termos.

Um exemplo da aplicação deste conceito à IG é o TGN. Este thesaurus tem como Termo Superior o mundo (*World*). Como Termos Específicos encontram-se os locais (*Places*) que estão organizados em diversas hierarquias relacionadas com contextos políticos, geográficos ou históricos (GETTY, 2005). Na figura 4.3 podem ver-se os resultados apresentados na pesquisa por "Braga". De salientar, a sua posição na hierarquia, com o Termo Genérico "Portugal" e "World" como Raiz.

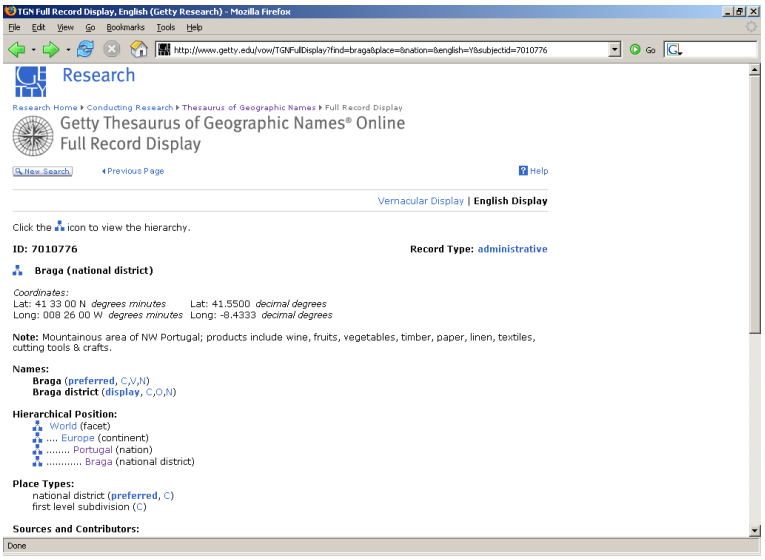


Figura 4.3: Definição do termo Braga, no TGN

4.6 Ontologia

Na secção anterior (4.5), apresentou-se o conceito de thesaurus que utiliza um conjunto de relações específicas para caracterizar um termo. Só pelo facto de estas relações serem específicas representam por si só uma limitação. Apresenta-se agora o conceito de Ontologia, ao qual se irá recorrer ao longo desta dissertação.

A ontologia descreve os conceitos existentes num mundo em particular através de uma qualquer relação binária que seja interessante criar (Librelotto, 2005).

Gruber define uma ontologia como sendo uma especificação explícita de uma conceptualização (Gruber, 1993). Uma conceptualização é uma interpretação estruturada de um mundo que as pessoas utilizam para pensar e comunicar sobre o mesmo (Borst, 1997).

Ao especificar uma conceptualização, ou seja, ao formalizar os conceitos e as relações que existem ou podem existir entre eles, estão a organizar-se sistematicamente percepções no âmbito de um domínio de conhecimento. Adicionalmente a formalização de uma conceptualização implica a existência de uma notação formal interpretável por máquinas e a sua aceitação por um grupo (Weissenberg and Gartmann, 2003). Consequentemente, o facto de uma ontologia ser bem definida e aceite por um grupo, não implica necessariamente que esta aceite por um outro grupo. Há que criar níveis de abstracção dentro das ontologias, para que se possa generalizar a sua aplicação. Em 1998, Guarino classificou as ontologias de acordo com o seu nível de generalidade (Guarino, 1998) (ver figura 4.4):

Ontologias de alto nível que descrevem conceitos gerais como tempo, espaço, coisa, acção, etc., independentemente do seu domínio. Estas ontologias de alto nível seriam utilizadas por uma grande comunidade de utilizadores, unificando a sua interpretação de forma abrangente e servindo como base à descrição de domínios mais específicos.

Ontologias de domínio e tarefa como o próprio nome indica, as ontologias de domínio são ontologias que especializam termos existentes nas ontologias de alto nível, mencionadas anteriormente, ou seja, descrevem domínios mais específicos como por exemplo a medicina, a informática e a geografia. As ontologias de tarefa, também são uma especialização das anteriores, mas descrevem tarefas ou actividades genéricas como diagnosticar, cartografar, etc.

Ontologias de aplicação descrevem conceitos dependentes, quer do domínio quer da tarefa, especializando as ontologias correspondentes. Ao ser o nível mais baixo, é consequentemente o mais específico.

A criação ou escolha das ontologias certas e a sua articulação não é só uma questão filosófica, mas também é um problema não trivial de engenharia. Esta problemática abrange a utilização das ontologias e a sua formalização, a representação dessas ontologias e a integração entre as diferentes ontologias e os domínios que estas descrevem (Wache et al., 2001).

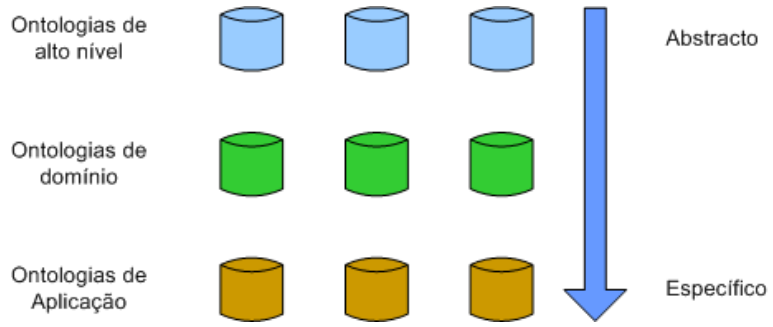


Figura 4.4: Classificação de ontologias quanto ao seu nível de generalidade

4.6.1 Web Ontology Language

Numa iniciativa do W3C, no âmbito dos trabalhos que este consórcio tem desenvolvido para a Semantic Web (W3C, 2001), surgiu a Web Ontology Language. A OWL pretende descrever formalmente o significado da terminologia utilizada nos documentos Web, isto para que seja possível dar às máquinas a capacidade de raciocinar sobre esses documentos.

No contexto da Semantic Web, as ontologias definem/descrevem os seguintes conceitos (W3C, 2004a):

Classes conceitos generalizados de um ou mais domínios.

Relações relações existentes entre classes.

Propriedades são as características que essas classes podem ter.

De acordo com o seu nível de expressividade, o W3C, define três variantes da OWL (W3C, 2004b), todas elas partilhando características da OWL Full (ver figura 4.5):

OWL Lite destina-se aos utilizadores cuja necessidade principal passa pelo uso de uma classificação hierárquica e de restrições simples. Possui restrições de cardinalidade apenas de zero ou um e é formalmente mais simples que a OWL DL.

OWL DL equilibra um máximo de expressividade com a garantia de ser possível tratá-la computacionalmente, num espaço de tempo finito. A OWL DL inclui todas as funcionalidades da linguagem OWL, mas apenas utilizáveis segundo determinadas restrições.

OWL Full destina-se aos utilizadores que necessitam de máxima expressividade e da liberdade sintáctica do RDF (*Resource Description Framework*[5]), sem garantias computacionais. Por exemplo, na OWL Full uma classe pode ser considerada simultaneamente como uma colecção de indivíduos e um indivíduo em si mesma.

[5] `http://www.w3.org/RDF`

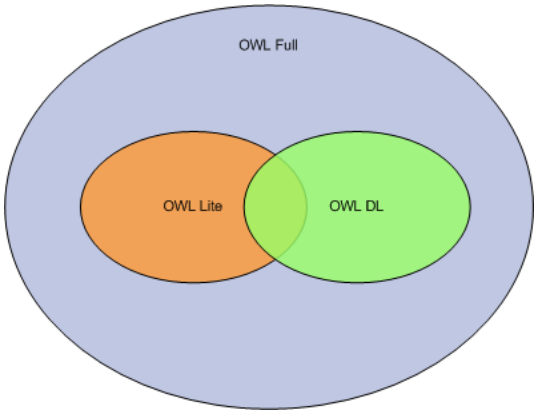


Figura 4.5: Sub-linguagens OWL

Ao tratar-se de uma linguagem XML, torna-a acessível e compreensível, quer por humanos, quer por máquinas. No entanto, facilmente se obtém vários milhares de linhas de código à medida que o domínio que se pretende descrever atinge níveis de detalhe elevados. Por esta razão têm surgido diversos projectos que visam a criação de editores OWL num formato visual. Exemplos desses projectos, são o OilEd (Bechhofer et al., 2001) (ver figura 4.6) e o Protegé (Grosso et al., 1999).

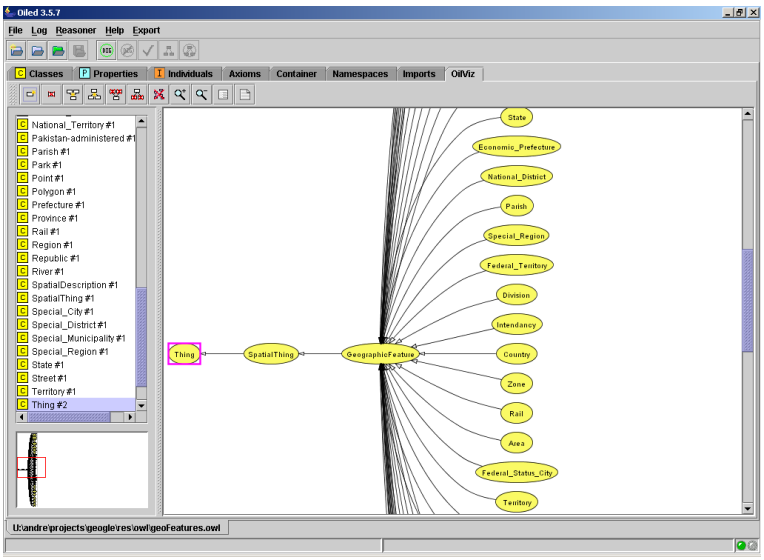


Figura 4.6: Edição de uma ontologia no Oiled

A problemática da integração de conteúdos na Web de proveniências diversas começou por ser parcialmente solucionada com o XML, que veio resolver a questão sintáctica uniformizando e permitindo a troca de informação num formato conhecido por todos. Posteriormente o XML Schema ampliou o âmbito da integração a um nível estrutural, permitindo não só definir a estrutura da informação, mas também proporciona a tipificação dos dados utilizados num documento XML. A OWL pretende alagar ainda mais a integração ao nível

do significado da informação.

Nos restantes capítulos, com base nas tecnologias e conceitos já apresentados, é proposto o reforço das capacidades semânticas dos Geo WS.

Capítulo 5

Enunciado do Problema

Nos capítulos anteriores, apresentaram-se alguns temas relacionados com a manipulação de IG em computador, nomeadamente as tecnologias dos Web Services (capítulo 2) e, em particular, dos Geo Web Services (capítulo 3). Acredita-se que a sua adopção generalizada pode contribuir para se criar uma extensa teia de serviços com uma mesma interface. Assim sendo, uma diversidade de clientes pode surgir, sem compromissos com formatos e serviços proprietários.

Abordaram-se também os conceitos relacionados com a representação e manipulação do conhecimento em computador (capítulo 4), para manipular a IG a um nível de abstracção superior. Infelizmente muita IG ainda é trabalhada ao nível do CAD (*Computer Aided Design*), sem a noção de entidades; apenas se manipulam desenhos formados por linhas, polígonos ou outras formas geométricas. Nos SIG, já se trabalham, representam e manipulam conceitos do mundo real, como rios, estradas, etc. Na nossa abordagem, queremos que estes conceitos do mundo real apareçam relacionados, de forma a, por exemplo, se poder oferecer a informação sobre "itinerários secundários", se o cliente solicitar informação sobre a "rede viária", já que os conceitos estão relacionados ao nível semântico. Estas relações, como veremos, têm que estar explicitadas em ontologias.

Embora os Geo Web Services se apresentem como uma solução para resolver vários problemas relacionados com a exploração da IG, subsistem ainda algumas dificuldades. Concretamente, todo o processo de identificação e escolha das fontes de informação carece de intervenção humana, não sendo possível (ainda) um cliente autonomamente descobrir e tirar partido de novas fontes de informação.

Neste capítulo começa-se por se apresentar um exemplo de sucesso de composição de vários Geo Web Services para fornecer informação sobre o Tsunami ocorrido no final de 2004. Isto para demonstrar que o método de composição dos serviços utilizados exigiu a interpretação humana dos documentos que os auto descrevem (retornados pela operação GetCapabilities). Discute-se a possibilidade de estes documentos serem processados autonomamente e interpretados por computador. Uma abordagem a este problema passa pela inclusão de meta-informação adequada, a qual exige o recurso a ontologias comuns.

5.1 Introdução

Os Geo WS têm grande potencial e a sua evolução prova que estes representam um desenvolvimento na partilha e disponibilização da IG. Ao explorar as suas características principais – composicionalidade e interoperacionalidade – podem proporcionar-se serviços em que a IG assume o papel principal, em detrimento do seu formato ou proveniência.

A arquitectura preconizada pelos WS, detalhada na secção 2.1.1, apontava como solução a criação e manutenção de um registo central com toda a informação sobre os WS. Para tal, foi desenvolvido o UDDI, referido na secção 2.4. Dado o diminuto sucesso deste registo UDDI (Sabbouh et al., 2001; Ambler, 2004) torna-se importante insistir em abordagens que potenciem a auto-descrição dos serviços, em vez de manter descrições externas dos mesmos. De alguma forma, manter um registo UDDI centralizado segue a mesma filosofia de um catálogo de uma biblioteca, em que se centraliza a meta-informação por limitações físicas.

Num ambiente Web, em que toda a informação está em formato digital, é possível e aconselhável manter a informação e a meta-informação juntas. Na abordagem que irá ser seguida nos capítulos 5 e 6, opta-se por desenvolver e explorar a adequada auto-descrição dos serviços, em vez de insistir na criação de catálogos centralizados e autónomos.

5.2 Tsunami – Caso de Estudo

Um bom caso de estudo que demonstra o que se tem vindo a reiterar acerca dos Geo WS ao longo desta dissertação, é uma solução desenvolvida pela *DM Solutions* [1] e pela Universidade de Otava que se aplica a uma situação bem recente do panorama mundial[2].

O terramoto que abalou a Ásia no final de 2004 e que afectou milhares de pessoas e centenas de localizações, obrigou a que diversas entidades de apoio interagissem entre si de forma a poderem prestar o apoio às vítimas desta catástrofe. Dada a dimensão do sinistro, a disponibilidade da IG é essencial para o planeamento das operações de salvamento e para a própria movimentação das equipas operacionais no terreno.

Foi então criado um portal[3] (ver figura 5.1) que reúne diversa informação geo-referenciada sobre o acidente. A IG disponível provém de diversas fontes; está localizada em diferentes servidores e eventualmente codificada em formatos heterogéneos. Toda esta infra-estrutura foi criada com base em serviços WMS e WFS de acordo com as especificações do consórcio OpenGIS.

O que este caso tem de interessante, não é só a diversidade da IG que o portal disponibiliza, mas é também o curto espaço de tempo em tudo foi realizado – cerca de 6 dias. Este exemplo, entre outros possíveis, demonstra o potencial dos Geo WS para solucionar

[1] Empresa fundada em 1998 com sede em Otava, no Canadá, fornecedora de soluções SIG vocacionadas para a Internet – `http://www.dmsolutions.ca`

[2] `http://www.cnn.com/2004/WORLD/asiapcf/12/27/asia.quake`

[3] `http://mapsherpa.com/tsunami`

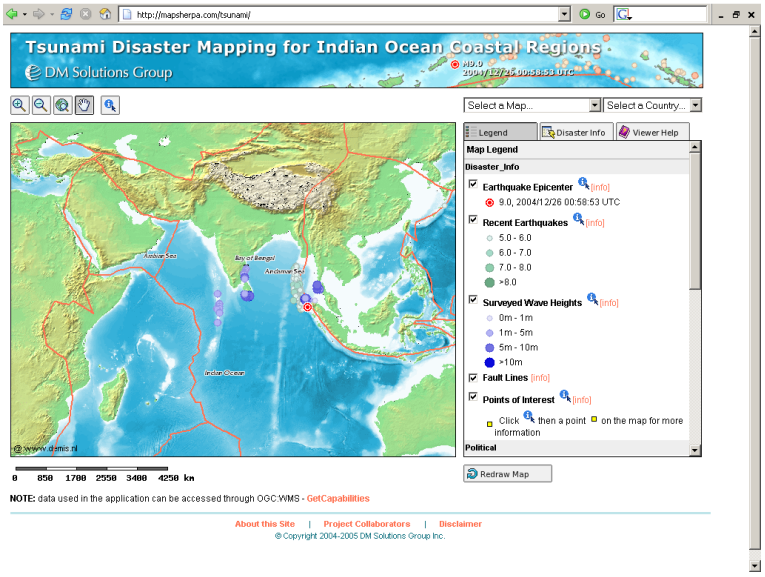


Figura 5.1: Portal com informação sobre o Tsunami que ocorreu em finais de 2004

os problemas sintácticos inerentes à interoperacionalidade da IG.

5.3 Desafios Semânticos

O caso de estudo apresentado é uma demonstração inequívoca que a utilização de Geo WS para disponibilizar IG traz vantagens importantes, na medida em que estes unificam a forma de aceder à mesma, não só no acesso a mapas através dos WMS, mas também no acesso a entidades através dos WFS.

Mesmo assim, e apesar destas vantagens constituírem um avanço significativo, existem ainda limitações de interpretação semântica da IG, limitações estas que serão demonstradas recorrendo a um exemplo.

No cenário apresentado na secção 5.2, conseguiu-se criar um serviço com IG proveniente de diversas fontes. Mas de que forma se garante que os WMS e WFS que o compõem contêm informação relevante? Mais, de que forma se combina e representa, para a Índia e Indonésia, a informação acerca de tremores de terra recentes, assumindo que a mesma provém de serviços diferentes?

Com base nesta segunda questão, apresenta-se um exemplo representativo da mesma, baseado no retorno da operação GetCapabilities de dois serviços distintos que retornam informação complementar sobre o mesmo conceito.

Listagem 5.1: Camada de um WMS que retorna informação sobre tremores de terra

```
...
<Layer queryable="0" opaque="0" cascaded="1">
  <Name>Earthquakes</Name>
  <Title>Indian Earthquakes</Title>
  <SRS>EPSG:4326</SRS>
  <MetadataURL type="TC211">
    <Format>text/html</Format>
    <OnlineResource/>
  </MetadataURL>
</Layer>
...
```

Listagem 5.2: Camada de um outro WMS que retorna informação sobre tremores de terra

```
...
<Layer queryable="0" opaque="0" cascaded="1">
  <Name>TremblementDeTerre</Name>
  <Title>Tremblement de terre</Title>
  <SRS>EPSG:4326</SRS>
  <MetadataURL type="TC211">
    <Format>text/html</Format>
    <OnlineResource/>
  </MetadataURL>
</Layer>
...
```

Só a interpretação humana pode ajudar a perceber se as camadas Earthquakes e TremblementDeTerre são equivalentes. Mesmo tendo o título sugestivo de "tremores de terra" (traduzido para Português) podem conter informação diversa; podem usar escalas diferentes para os tremores de terra; podem usar só a informação do epicentro (ponto) ou de toda a zona afectada (polígono), etc. O humano aparece como intérprete da informação, eliminando assim dúvidas e ambiguidades, ou, por outro lado, identificando a potencial equivalência e complementaridade entre as fontes de informação ou serviços.

Conclui-se que a falta informação semântica sobre os dados é colmatada pela introdução do humano na cadeia. Este facto leva a que não seja atingida a verdadeira interoperacionalidade entre Geo WS.

5.4 Inclusão de meta-informação

A abordagem mais pragmática para esta questão seria a inclusão de informação semântica associada aos serviços e dados que estes fornecem, para assim ultrapassar esta limitação dos Geo WS.

Pode começar-se por utilizar um recurso já previsto pela operação GetCapabilities, na declaração das camadas através do elemento Layer – a utilização do elemento KeywordList . Desta forma, utilizar-se-ia um conjunto de palavras-chave que iriam permitir marcar cada uma das camadas fornecidas com o seu significado, conforme se mostra nos exemplos seguintes:

Listagem 5.3: A mesma camada apresentada na listagem 5.1, explorando a utilização de palavras-chave.

```
...
<Layer queryable="0" opaque="0" cascaded="1">
  <Name>Earthquakes</Name>
  <Title>Indian Earthquakes</Title>
  <KeywordList>
    <Keyword>earthquake</Keyword>
    <Keyword>sismic activity</Keyword>
  </KeywordList>
  <SRS>EPSG:4326</SRS>
  <MetadataURL type="TC211">
    <Format>text/html</Format>
    <OnlineResource/>
  </MetadataURL>
</Layer>
...
```

Listagem 5.4: A mesma camada apresentada na listagem 5.2, explorando a utilização de palavras-chave

```
<Layer queryable="0" opaque="0" cascaded="1">
  <Name>TremblementDeTerre</Name>
  <Title>Tremblement de terre</Title>
  <SRS>EPSG:4326</SRS>
  <KeywordList>
    <Keyword>earthquake</Keyword>
    <Keyword>sismic activity</Keyword>
  </KeywordList>
  <MetadataURL type="TC211">
    <Format>text/html</Format>
    <OnlineResource/>
  </MetadataURL>
</Layer>
...
```

Seria de esperar que todas as outras camadas também aparecessem marcadas com o respectivo significado. No entanto, ao falar em semântica coloca-se também a questão do domínio e do significado das palavras-chave incluídas...

5.5 Ontologias precisam-se!

Para que estas palavras-chave funcionassem, teria que se estabelecer um vocabulário comum, onde ocorressem palavras cujo significado estivesse bem entendido, o que não seria tarefa fácil, dado a abrangência da comunidade de utilizadores de IG. Além disso, seria de todo conveniente que fossem estabelecidas as relações entre os termos desse vocabulário que permitissem inferir, por exemplo, que um tremor de terra é um abalo sísmico.

Para tal, poder-se-ia recorrer a um thesaurus, com a capacidade de relacionar todos os conceitos capturados pela IG. No entanto, estes são estritos nas relações que estabelecem

entre os termos que descrevem, o que, como já foi mencionado na secção 4.5, é uma limitação.

Se o objectivo é atingir a interoperacionalidade, há que trabalhar no sentido da descrição formal do conhecimento presente nos seus diversos domínios, e isso pode ser obtido através da utilização de ontologias (Wiederhold, 1994).

Para recorrer à utilização de ontologias, em vez de thesaurus, é necessário resolver questões a montante dos aspectos tecnológicas que têm sido discutidos ao longo desta dissertação, que se levantam fundamentalmente a dois níveis (Araújo and Rocha, 2004):

Metafísico Como conceptualizar o mundo real e como obter a percepção sobre o mesmo?

Ontológico Que fenómenos têm significado e qual é esse significado?

Esta inclusão de informação semântica, processável por máquinas, vem alargar a abrangência da interoperacionalidade para o âmbito do significado da informação. As ontologias são essenciais para a interoperacionalidade entre sistemas que operam em ambientes heterogéneos (Farquhar, Fikes, and Rice, 1996).

Na secção 4.6 abordou-se a questão da engenharia de ontologias no contexto da representação de conhecimento. Referiu-se também a necessidade de criação de ontologias capazes de fornecer vários níveis de abstracção de forma a possibilitar a reutilização e composição de ontologias. Isto tem a finalidade de ser possível construir uma arquitectura ontológica capaz de descrever determinados domínios do conhecimento de forma abrangente e flexível.

Obter a interoperacionalidade semântica em sistemas de informação através da utilização de ontologias, exige, no limite, a interoperacionalidade entre as ontologias que o compõem. A composição de ontologias foi estudada em (Wiederhold and Jannink, 1999).

Através a utilização ontologias pretende-se eliminar as ambiguidades semânticas. Em (Goh, 1997) enumeram-se as três principais causas para essa ambiguidade:

Conflitos denominativos quando o mesmo termo pode ter significados diferentes. Estes identificam-se aquando da utilização de homógrafos ou homónimos. Exemplificando, a palavra "carta" pode ser utilizada para significar "mapa" ou "missiva".

Conflitos de escala e unidade relacionados com a adopção de unidades de medida ou escalas. Por exemplo, na comparação de distâncias, a utilização de metros ou milhas.

Conflitos de equidade ocorrem quando dois elementos têm aparentemente o mesmo significado, não havendo conflitos denominativos, mas na realidade diferem, como é o da informação que tem uma componente temporal intrínseca. Por exemplo, a densidade populacional pode ter diferentes significados ou interpretações dependendo do ano a que se refere.

Desta feita, o desenvolvimento de ontologias não é um processo trivial, que exige a cristalização de consensos sobre as áreas a que respeitam. No âmbito da IG, também não tem sido fácil a aceitação de consensos sobre como modelar o mundo que nos rodeia. Na secção 5.6 aborda-se a criação de ontologias no domínio dos SIG.

5.6 Ontologias nos Sistemas Informação Geográfica

A utilização de ontologias em SIG foi objecto de estudo em (Smith and Mark, 1998), onde se atestam as vantagens da utilização de ontologias no domínio da geografia, contribuindo para um melhor entendimento desse domínio:

- Permitindo perceber como é que diferentes comunidades de utilizadores trocam e fazem uso da IG.
- Ajudando a identificar distorções na percepção do mundo, percepção essa que se relaciona o nosso processo cognitivo.
- Normalizando a conceptualização do mundo, fazendo com que se trabalhe numa base de entendimento comum.

5.6.1 Ontologias para a Informação Geográfica

Existem já alguns trabalhos, terminados ou em curso, cujo principal objectivo é a criação de ontologias para descrição de determinados domínios da IG. De todas estas ontologias destacam-se algumas:

SOUPA *Standard Ontology for Ubiquitous and Pervasive Applications* (SOUPA) (Chen et al., 2004), é um conjunto de ontologias que vêm sendo desenvolvidas pelo grupo *Semantic Web In Ubicomp*[4]. Estas ontologias, escritas em OWL, encontram-se divididas em dois grandes grupos:

- SOUPA Core: que descreve vocabulários genéricos que se pretendem universais para aplicações móveis e ubíquas.
- SOUPA Extension: descreve vocabulários específicos para algumas dessas aplicações.

COBRA ONT conjunto de ontologias, desenvolvidas em OWL, para serem utilizadas por aplicações ou sistemas sensíveis ao contexto (Chen, Finin, and Joshi, 2004b). Estas ontologias estão no âmbito do projecto *Context Broker Architecture* (CoBrA)[5] que é uma *framework* para suportar sistemas baseados em contexto, que permite criar salas inteligentes (Chen, Finin, and Joshi, 2004a).

geoOntologies trata-se de um projecto de onde resultaram várias ontologias em OWL que abrangem conceitos ou sub-domínios da IG, são elas (Clark, Hiramatsu, and Reitsma, 2004):

- *geoFeatures*: esta ontologia descreve entidades geográficas e as suas características espaciais.

[4] `http://pervasive.semanticweb.org`
[5] `http://cobra.umbc.edu`

- *geoRelations*: ontologia que descreve relações meteorológicas, topológicas e de distância.
- *geoCoordinateSystems*: apesar de estar num estado embrionário, esta ontologia pretende descrever os sistemas de coordenadas existentes.

SWEET é acrónimo de *Semantic Web for Earth and Environmental Terminology*. Trata-se de um conjunto de ontologias, escritas em OWL e disponíveis em `http://sweet.jpl.nasa.gov/sweet`, com o propósito de descreverem a ciência do sistema terrestre (Raskin, 2004).

Neste capítulo demonstrou-se a dificuldade, baseada num pequeno exemplo, de integrar e sobrepor informação complementar sobre tremores de terra, provenientes de servidores distintos. Argumenta-se que é indispensável a intervenção humana para que os clientes descubram e explorem autonomamente novos servidores. Mostrou-se ainda a necessidade de se criarem consensos que possam ser materializados em ontologias, mas que é algo difícil e demorado de se obter. Foram feitas algumas referências a trabalhos anteriores com o propósito de desenvolver e disponibilizar ontologias no domínio da IG.

Capítulo 6

Resolução do Problema

Como forma de contribuir para a resolução destas questões optou-se por seguir em duas frentes. Uma que consistiu na criação de um motor de busca com a capacidade de responder a inquéritos com base numa camada ontológica e outra que foi a participação e acompanhamento de uma experiência realizada pelo consórcio OpenGIS – *Geospatial Semantic Web Interoperability Experiment* (GSW IE).

Caminhar por estes dois caminhos distintos tornou possível, ao mesmo tempo, estudar e partilhar ideias e conceitos duma forma cooperativa através da GSW IE, e, adicionalmente, provar um conceito.

6.1 Abordagens à Semântica

Apesar do consórcio OpenGIS ter iniciado esta experiência apenas em 2005, já existia algum frenesi na comunidade científica em torno desta questão, pelo que têm sido apresentadas algumas propostas e/ou ideias que abordam esta temática.

Lemmens apresentou uma proposta para a descrição de Serviços Baseados em Localização (SBL), que abordava a problemática da localização de SBL relevantes (Lemmens and de Vries, 2004). Para essa descrição, foi criada uma ontologia de localização que descreve alguns conceitos básicos da informação geográfica.

Em 2004, Hiramatsu apresentou um caso de estudo que demonstrou a capacidade de se criarem afinidades entre os conceitos da Semantic Web e a necessidade de informação semântica na IG (Hiramatsu and Reitsma, 2004).

Paralelamente ao seu envolvimento na GSW IE, Kolas propõe uma solução arquitectural baseada num WFS semântico que interage com uma série de ontologias organizadas hierarquicamente (Kolas, Hebeler, and Dean, 2005). Esta proposta, está naturalmente em linha com o que tem sido o trabalho da GSW IE e reforça ainda a necessidade de normalização de WS semânticos.

A unificação entre semântica e WS foi também objecto de estudo de Martin, em 2004,

que defende a utilização de uma *Ontology Web Language for Services*[1] (OWL-S) que permita, em conjunto as normas existentes, complementar a descrição de WS com informação semântica (Martin et al., 2004).

6.2 Caso de estudo/Demonstração – Geogle

6.2.1 Introdução

No âmbito desta dissertação, procurou-se criar um caso de estudo (ou prova de conceito) que fosse um bom ponto de partida para a concretização de todas estas ideias relacionadas com WS semânticos. Daí o desenvolvimento do **Geogle**, o nome "Geogle" (ver figura 6.2) advém da combinação entre "Geografia" e "Google"[2]. Fundamentalmente o Geogle é um motor de busca para Geo WS. Este motor de busca combina a simplicidade do Google com a pesquisa em Geo WS. Esta pesquisa que pode ser feita de forma meramente sintáctica, isto é, verificando correspondências textuais entre os termos introduzidos pelo utilizador e os dados existentes nas descrições dos WS, ou através de inferências – sobre uma base de conhecimento – que fornecem resultados provenientes de uma análise semântica num domínio do conhecimento bem definido e especificado através de ontologias. Esta segunda forma de pesquisa, que consiste em pesquisas que recorrem a inferências sobre uma base de conhecimento, definido por uma ontologia, é um contributo inovador desta tese.

Na secção 3 mostrou-se que os Geo WS são caracterizados por um conjunto de atributos textuais, como por exemplo, o nome, o título, o *abstract*, uma lista de palavras-chave, entre outros. Mas também se caracterizam pelas suas camadas e respectivos atributos que as compõe. Alguns destes necessitam de processamento extra para serem interpretados, como são exemplo os atributos que reflectem dimensão. Porque as camadas que um serviço disponibiliza, estão organizadas de forma hierárquica, existe entre elas uma relação hierárquica que não pode ser esquecida.

O Geogle, ao contrário das outras soluções que foram mencionadas, concentra-se na descrição da informação que o WS disponibiliza, por oposição à descrição das suas *interfaces*. É desta forma que pode fazer inferências em qualquer Geo WS cujo formato esteja conforme as normas do consórcio OpenGIS e sem a necessidade de configurações adicionais no caso de se pretender adicionar outros Geo WS.

Apesar de ser um motor de busca que pode ser utilizado por humanos através de um navegador comum, o Geogle é também um WS, podendo ser utilizado por outros WS sem a intervenção humana. O diagrama de casos de uso na figura 6.1 ilustra as principais interacções entre os actores e o sistema.

A lógica de funcionamento do Geogle centra-se numa óptica orientada à entidade, isto é, permite fazer uma pesquisa no sentido de retornar todos os serviços que contêm informação sobre uma entidade. Por exemplo:

[1] `http://www.daml.org/services/owl-s`
[2] `http://www.google.com`

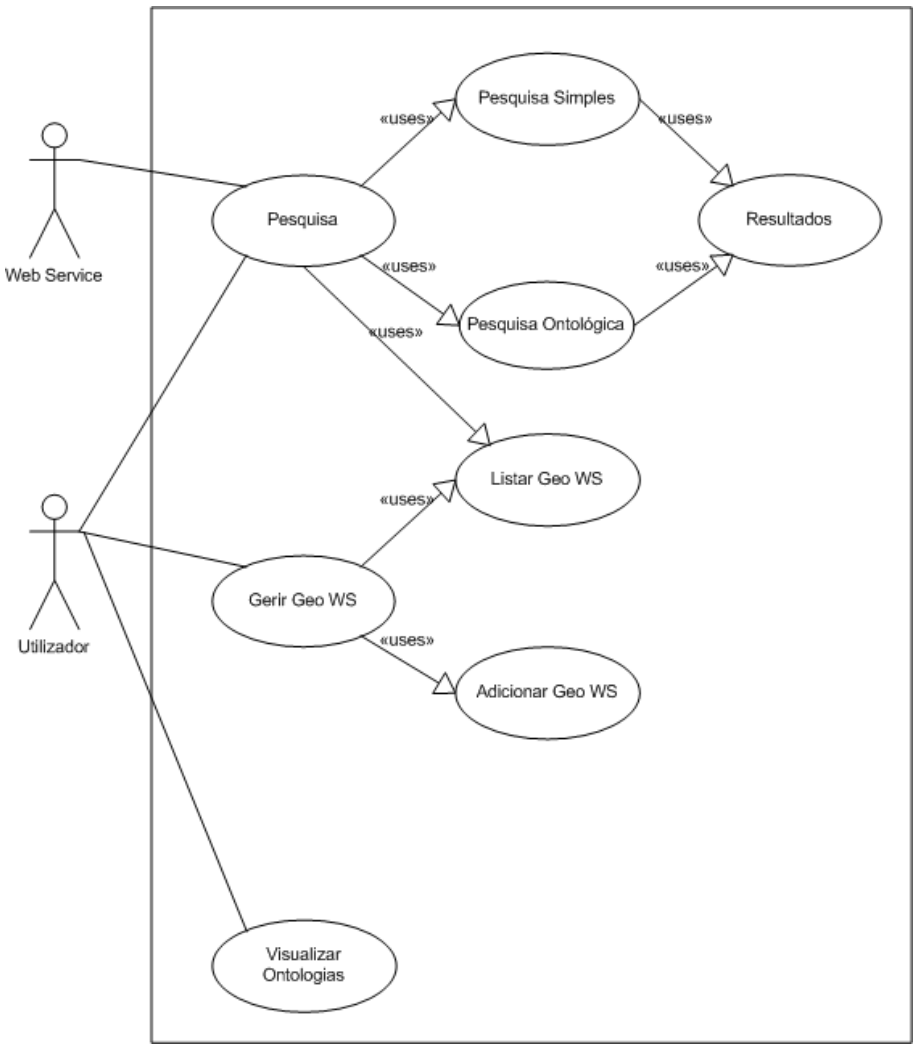


Figura 6.1: Geogle – Casos de uso

- Todos os serviços com informação acerca de **vias**.
- Todos os serviços com informação acerca de **espaços verdes**.

Consoante indicação do actor, o sistema efectua um de dois tipos de pesquisa, a pesquisa ontológica, que tem como base uma ou mais ontologias, e uma pesquisa textual tradicional, baseada apenas em palavras chave, a que se chama uma pesquisa simples.

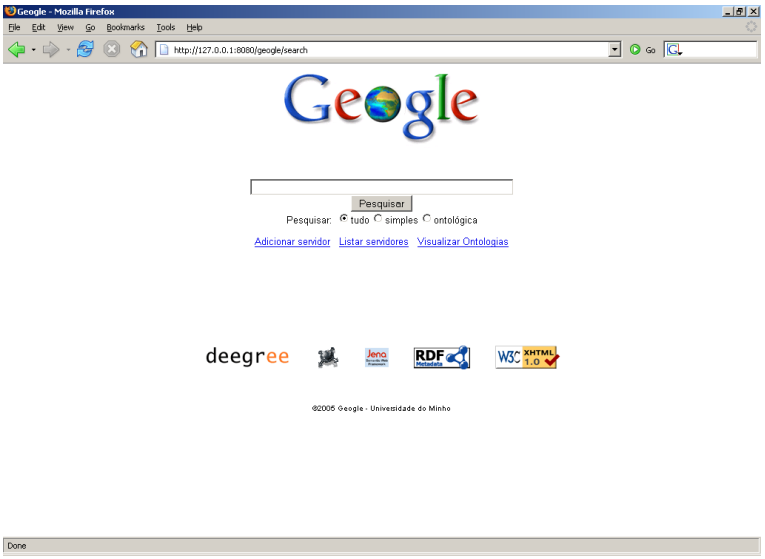


Figura 6.2: Geogle

6.2.2 Arquitectura

Na figura 6.3 ilustra-se a arquitectura do Geogle, que na sua essência é composta por três componentes fundamentais:

Cliente disponibiliza uma interface gráfica (previamente ilustrada na figura 6.2 que executa pedidos ao servidor. Estes pedidos são constituídos por alguns parâmetros e por uma lista de palavras a pesquisar, indicados pelo utilizador.

Servidor trata-se de um *servlet* que coordena a interacção do utilizador. Ao receber a lista de parâmetros a pesquisar, o servidor processa o comando respectivo, e inicia a sua execução.

Base de conhecimento constituída por uma ou mais ontologias escritas em OWL que descrevem entidades geográficas. Estas ontologias foram criadas tendo por base aquelas que foram desenvolvidas pelo grupo *Semantic geoStuff* [3], já referidas na secção 5.6.1.

[3] `http://www.mindswap.org/2004/geo/geoStuff.shtml`

Inventário de Geo WS uma base de dados com os serviços que vão ser pesquisados pelo Geogle.

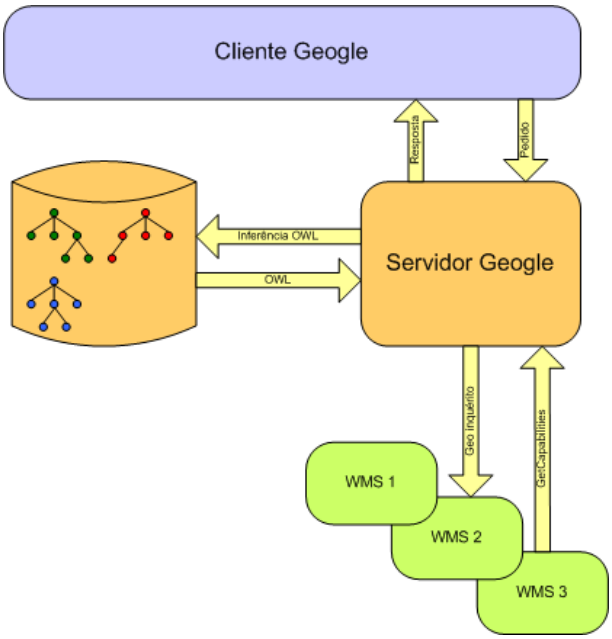


Figura 6.3: Geogle – Arquitectura de pesquisa

O servidor, independentemente da forma como é acedido, responde a pedidos. Um pedido traduz-se sempre numa pesquisa, e quer seja esta ontológica ou simples, vão ser consultados todos os Geo WS inventariados que o Geogle guarda em base de dados de forma a obter a descrição XML do GetCapabilities respectivo. Se a pesquisa solicitada pelo cliente for ontológica, há que inferir os resultados tendo em conta a base de conhecimento existente. O algoritmo de pesquisa será detalhado posteriormente na secção 6.2.5.

6.2.3 Dependências

O Geogle foi desenvolvido utilizando algumas bibliotecas de referência no domínio dos Geo WS e OWL, isto porque, ao utilizar software aberto, foi possível não só ter acesso ao código fonte das referidas bibliotecas, mas também trabalhar em conjunto com os grupos que as desenvolvem para melhoramentos e detecção/correcção de erros.

Na figura 6.4 pode ver-se um diagrama de pacotes e as suas dependências externas e algumas internas:

patterns Trata-se de um pacote de utilitários, desenvolvido no âmbito desta dissertação, que implementa e facilita a utilização de padrões de software (Gamma et al., 1994).

log4j é um projecto de software aberto que permite o controlo, flexível, de cada saída para

registo de *logs*[4]. Entre outras funcionalidades, permite a activação/configuração dos logs em tempo de execução.

servlet tecnologia baseada na linguagem Java que possibilita a criação de páginas Web dinâmicas e o desenvolvimento de aplicações do lado do servidor.

deegree projecto de software aberto que oferece uma plataforma de criação e/ou acesso a infra-estruturas conformes às normas do consórcio OpenGIS[5]. Uma vez que esta solução segue uma norma bem definida, pode ser utilizada com outros serviços desenvolvidos segundo essa norma.

jena projecto aberto de software para manipulação de ontologias escritas em OWL. Este projecto está a ser desenvolvido pelo *HP Labs Semantic Web Group*[6], no âmbito do *HP Labs Semantic Web Programme* e possui, entre outras, funcionalidades como:

- API para manipulação de ontologias nos formatos RDF, Daml+Oil e OWL.
- Suporte para inferência de ontologias OWL.

httpclient biblioteca Java desenvolvida pela Apache para criação de software que utiliza o protocolo HTTP para comunicação. O Geogle utiliza pedidos HTTP para efectuar chamadas aos WS que estão registados para pesquisa.

xerces parser XML, desenvolvido pela Apache, para converter um documento XML num objecto DOM.

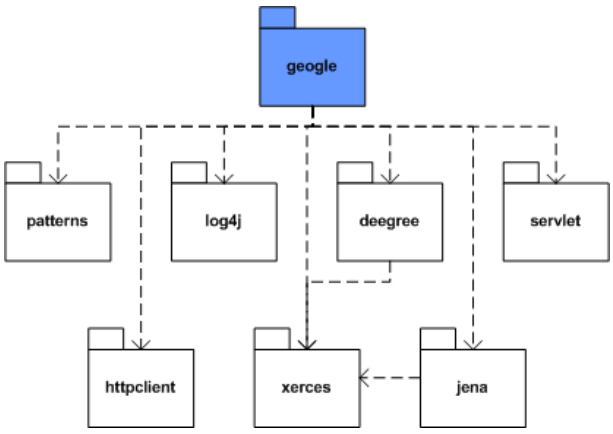


Figura 6.4: Geogle – Diagrama de dependências

6.2.4 Estrutura de classes

Para um nível mais detalhado da arquitectura do Geogle, recorre-se a um diagrama de classes (ver figura 6.5). Numa primeira observação, consegue ver-se que se trata de uma

[4] `http://logging.apache.org/log4j`
[5] `http://deegree.sourceforge.net/`
[6] `http://www.hpl.hp.com/semweb/`

arquitectura relativamente simples, na medida em que não existem relações de grande complexidade entre as classes que compõem o Geogle.

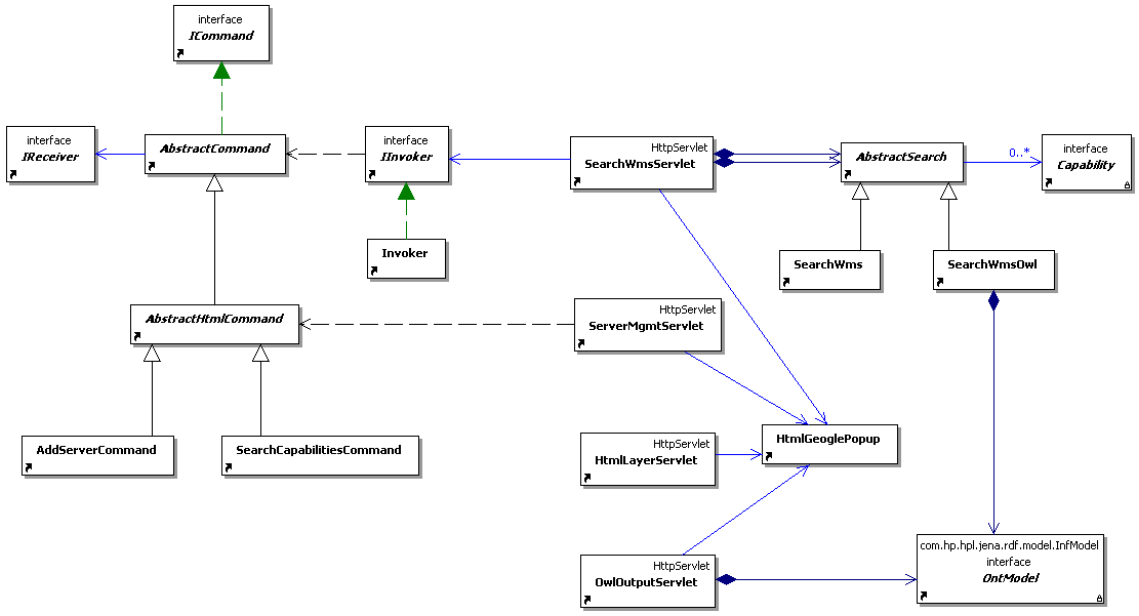


Figura 6.5: Geogle – Diagrama de classes

Internamente o Geogle divide-se em dois pacotes principais:

patterns este pacote implementa uma série de padrões de software (Gamma et al., 1994), com base nas implementações Java sugeridas em (Stelting and Maassen, 2002).

geogle contém toda a lógica do Geogle, nomeadamente, interface com o utilizador, pesquisa simples, pesquisa ontológica, *cache* de serviços e respectivas *capabilities*.

Em detalhe, o pacote *patterns* é composto por várias classes e interfaces, que implementam, ou facilitam a implementação, de vários padrões de software. O Geogle utiliza apenas a *Command Pattern* (Gamma et al., 1994), implementada pelas seguintes classes e *interfaces*:

ICommand contém apenas a assinatura dos métodos a serem invocados pelo invoker, e que se referem à execução do comando.

AbstractCommand um comando abstracto, que implementa a interface ICommand. Os comandos concretos devem ser implementados dentro dos pacotes clientes.

IReceiver um objecto que implementa esta interface é o destino do comando, ou o objecto no qual depende a execução de um comando.

IInvoker um objecto que implemente esta interface, invoca o comando.

Invoker implementação concreta da interface IInvoker.

AbstractHtmlCommand implementação abstracta dum comando, contém lógica para tratar e retornar informação num formato HTML.

AddServerCommand este comando é uma implementação concreta de AbstractHtmlCommand, cuja implementação contém a lógica para adição de um novo servidor ao Geogle, de forma a que este seja carregado em *cache*.

SearchCapabilitiesCommand este comando é uma implementação concreta de AbstractHtmlCommand e contém a informação necessária para efectuar um determinado tipo de pesquisa. Ou ontológica, ou simples. Este comando contém uma dependência implícita para a classe AbstractSearch.

O pacote *geogle* é composto por todas as classes que contêm a lógica de execução da aplicação, quer ao nível de interacção com o utilizador, quer ao nível da lógica de pesquisa. Este pacote é composto por quatro *servlets* que servem de ponto de acesso a toda a solução:

SearchWmsServlet disponibiliza a interface gráfica com utilizador, e processa os parâmetros da pesquisa. Com base nesses parâmetros, é construído um comando com a lógica correspondente à pesquisa. Esse comando é executado por esta classe e o seu resultado mostrado ao utilizador. Esta classe contém duas referências para as classes SearchWms e SearchWmsOwl. A primeira contém a lógica de pesquisa simples, e a segunda contém a lógica para a pesquisa ontológica, mantendo uma referência para o respectivo modelo ontológico (OntModel).

ServerMgmtServlet o Geogle contém uma lista de servidores mantidos em memória, bem como o retorno respectivo da operação getCapabilities. É possível, em tempo de execução, gerir essa lista de servidores, sendo que pedidos com este fim são reencaminhados para esta classe.

HtmlLayerServlet para converter uma camada em HTML, foi criada esta classe. Quando o utilizador pretende visualizar os detalhes de uma determinada camada, o pedido é reencaminhado para esta classe, que converte os dados de XML para HTML e os disponibiliza.

OwlOutputServlet para visualizar relações presentes nas ontologias que suportam o Geogle, foi criada esta classe, que, de forma recursiva, faz a pesquisa no grafo, inferindo classes, classes equivalentes e indivíduos. Essa informação é processada para posterior construção de uma árvore em DHTML para ser navegada e visualizada pelo utilizador.

6.2.5 Lógica de Pesquisa

O funcionamento do Geogle compõe dois tipos de pesquisa: pesquisa simples, e pesquisa ontológica. Esses tipos de pesquisa seguem uma lógica que é agora detalhada em alto nível:

1. Pesquisa simples:
   (a) Processamento dos dados inseridos pelo utilizador.
   (b) Pesquisa pelas descrições textuais referentes às camadas de um serviço. São retornadas as camadas dos Geo WS cujos elementos contêm texto total ou parcialmente igual ao inserido pelo utilizador.
   (c) Apresentação dos resultados.
2. Pesquisa ontológica:
   (a) Processamento dos dados inseridos pelo utilizador.
   (b) Inferência numa ontologia de classes e instâncias, que é feita nas seguintes fases:
      i. Construção de um mapa em memória com as classes presentes na base de conhecimento, inferindo as suas classes equivalentes e associando-as a cada uma das suas bases. O mapa é construído na forma:
      $classe \Rightarrow equivalente_1, equivalente_2, equivalente_n$.
      ii. Itera-se sobre os elementos do mapa criado anteriormente por forma a verificar semelhanças entre o texto a pesquisar e a informação existente para cada uma das classes. De salientar que se uma das classes é inferida como objecto da pesquisa, então todas as suas equivalentes são também objecto da pesquisa. No fim da execução deste passo, está construída uma lista com todas as classes da ontologia que serão utilizadas para compor o resultado da pesquisa.
      iii. Percorrendo de forma recursiva cada camada de todos os Geo WS, vai-se verificar se existe correspondência entre pelo menos uma das classes presentes na lista previamente construída. Se sim, essa camada é retornada no resultado da pesquisa.
   (c) Apresentação dos resultados.

Em ambos os tipos de pesquisa, a primeira fase consiste no processamento dos dados introduzidos pelo utilizador de forma a fazer a parametrização do comando correspondente à pesquisa a efectuar.

A segunda fase, difere consoante o tipo de pesquisa. Para a pesquisa simples é feita uma pesquisa sucessiva por todos os WS e para cada uma das suas camadas pelas palavras introduzidas pelo utilizador.

No caso da pesquisa ontológica, como a própria designação indica, tem por base uma ontologia de classes codificada em OWL. Para isso, são analisadas as palavras introduzidas pelo utilizador que são cruzadas com as classes existentes no modelo. Os resultados são então baseados na semelhança entre o introduzido pelo utilizador e as classes presentes na ontologia.

Na terceira fase são compostos os resultados em HTML que serão apresentados ao utilizador.

Na figura 6.6 ilustra-se o processamento dos pedidos de pesquisa ontológica, simples e adição de um novo servidor por parte do Geogle:

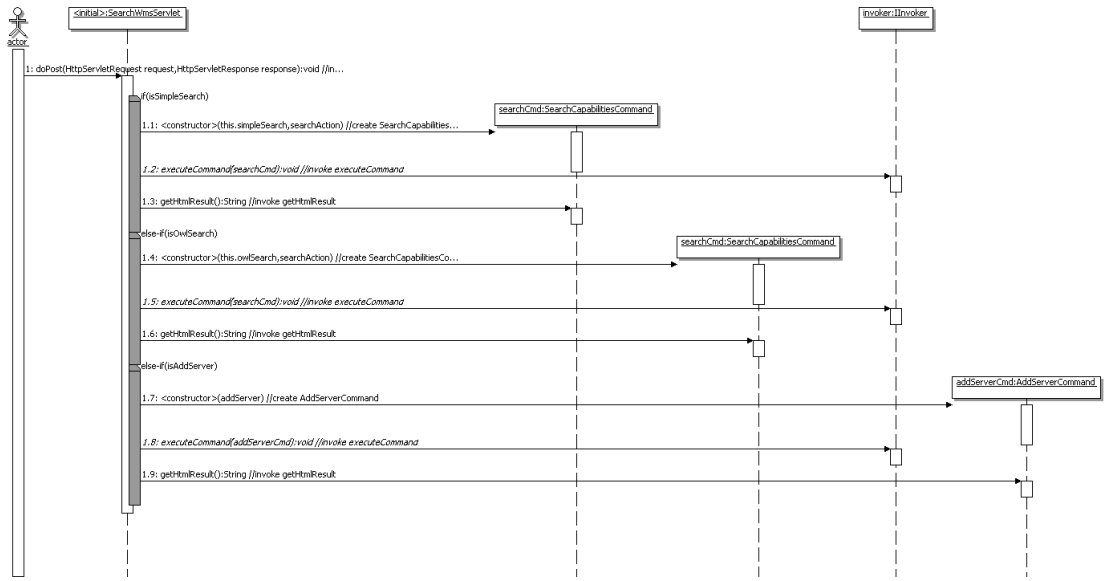


Figura 6.6: Geogle – Encaminhamento de pedidos de pesquisa e gestão de servidores

1. Um pedido começa por ser feito via HTTP POST à classe SearchWmsServlet.
2. Os parâmetros passados no pedido são analisados e se se tratar de uma pesquisa simples:
   (a) Constrói-se o comando correspondente – SearchCapabilitiesCommand. É feita a parametrização do comando com uma implementação concreta de AbstractSearch que, para este caso, é uma instância da classe SearchWms.
   (b) Executa-se o mesmo via IInvoker.
   (c) Processa-se o resultado (sucesso ou insucesso) em HTML para ser devolvidos.
3. Por outro lado, se se tratar de uma pesquisa ontológica:
   (a) Constrói-se o comando correspondente – SearchCapabilitiesCommand. É feita a parametrização do comando com uma implementação concreta de AbstractSearch que, para este caso, é uma instância da classe SearchWmsOwl.
   (b) Executa-se o mesmo via IInvoker.
   (c) Processam-se os resultados em HTML para serem devolvidos.
4. Para a adição de um novo servidor:
   (a) Constrói-se o comando correspondente – AddServerCommand.
   (b) Executa-se o mesmo via IInvoker
   (c) Processam-se os resultados em HTML para serem devolvidos.

6.2.6 Plataforma

O Geogle foi desenvolvido em Java e é, por isso, independente do sistema operativo. É necessário um adaptador específico para o tornar acessível via HTTP e, para isso foi utilizado o Tomcat 5.5. Para a edição e alteração de ontologias foram utilizados dois editores, o XML Spy e o OilEd.

6.2.7 Funcionalidade

Para explicar a funcionalidade do Geogle recorre-se a um exemplo que será composto pela utilização de todas as funcionalidades principais do motor de busca.

Pretende-se com a execução deste exemplo o seguinte:

1. Introduzir um WMS disponível na Universidade do Minho na base de dados do Geogle.
2. Visualizar as classes, indivíduos e relações da base de conhecimento.
3. Efectuar uma pesquisa com o objectivo de retornar todas as camadas com informação sobre "Edifícios", numa pesquisa simples e ontológica. Isto para que se possa comparar resultados.

Gestão de novos servidores

Para a gestão de novos servidores em tempo de execução, o utilizador tem a opção *Adicionar Servidor* (ver figura 6.7), para adicionar um novo servidor, ou a opção *Listar Servidores* (ver figura 6.8), para listar todos os servidores suportados pelo Geogle. Relativamente à listagem dos servidores configurados, o resultado é apresentado sob a forma de um URL que retorna o documento XML correspondente ao pedido GetCapabilities para o servidor em questão.

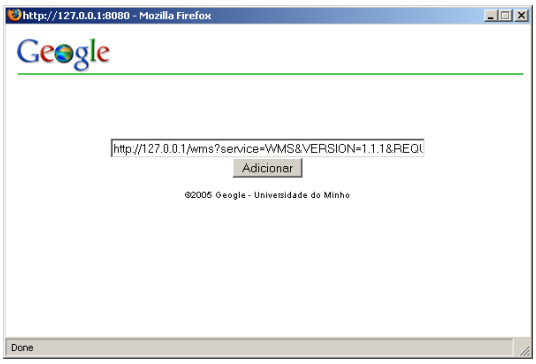


Figura 6.7: Geogle – Adição de um novo servidor em tempo de execução, indicando o URL que permite obter o documento XML retornado pela operação GetCapabilities

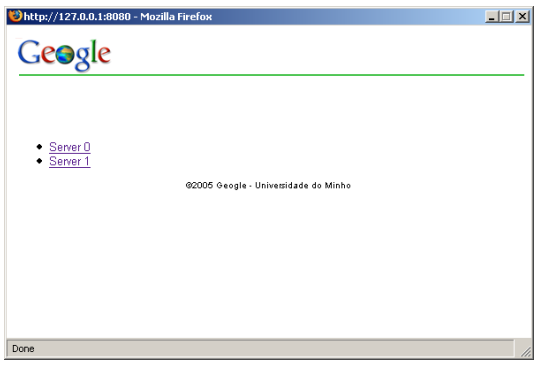


Figura 6.8: Geogle – Listar servidores actualmente configurados

Visualização de ontologias suportadas

A visualização das ontologias suportadas é feita recorrendo a procuras na base de conhecimento que permitem obter as classes, as classes equivalentes, os indivíduos e as relações hierárquicas entre eles.

O utilizador pode obter uma vista hierárquica da base de conhecimento do Geogle utilizando a opção *Visualizar Ontologias*. Um exemplo do resultado dessa operação pode ser visto na figura 6.9. Neste exemplo pode-se ver o topo da hierarquia de classes, a classe *SpatialThing*, as instâncias da classe *City*, e as classes equivalentes à classe *Lake*.

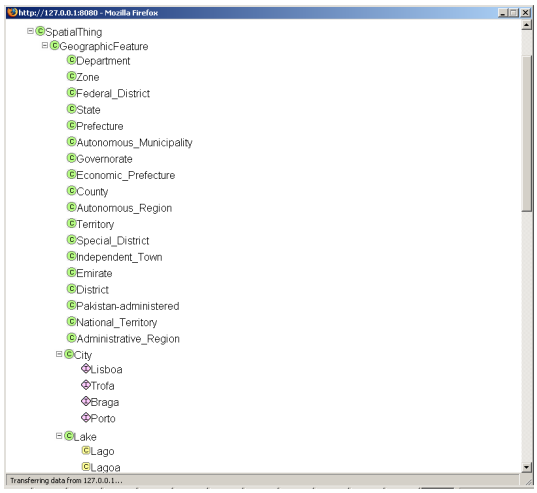


Figura 6.9: Geogle – Visualizar Ontologias

As classes e instâncias mostradas na figura 6.9, são codificadas em OWL, um exemplo que descreve caminhos-de-ferro aparece na listagem seguinte. As classes são utilizadas para descrever as entidades (por exemplo, caminhos-de-ferro), os conceitos equivalentes são codificados utilizando classes equivalentes (por exemplo, a classe Caminhos−de−Ferro tem uma relação de equivalência com a classe Rail). Instâncias das respectivas classes (por exemplo, Eixo-Porto-Aveiro) são efectivamente instâncias OWL.

Listagem 6.1: Código OWL que descreve a entidade caminhos-de-ferro

```
...
  <owl:Class rdf:ID="Rail">
    <rdfs:label>Rail</rdfs:label>
    <rdfs:subClassOf rdf:resource="#GeographicFeature"/>
  </owl:Class>

  <owl:Class rdf:ID="Caminhos-de-ferro">
    <owl:equivalentClass rdf:resource="#Rail"/>
  </owl:Class>

  <Rail rdf:ID="Eixo-Porto-Aveiro"/>
  <Rail rdf:ID="Eixo-Porto-Braga"/>
  <Rail rdf:ID="Eixo-Porto-Guimaraes"/>
  <Rail rdf:ID="Eixo-Porto-Caide-Marco-de-Canaveses"/>
  <Rail rdf:ID="Linha-Coimbra-Figueira-da-Foz"/>
  <Rail rdf:ID="Linha-de-Sintra"/>
  <Rail rdf:ID="Linha-de-Azambuja"/>
  <Rail rdf:ID="Linha-de-Cascais"/>
  <Rail rdf:ID="Linha-de-Sado"/>
...
```

Pesquisas e análise de resultados

Por fim apresenta-se o resultado de uma pesquisa simples e ontológica por "Edificios". Os resultados dessa pesquisa aparecem ilustrados na figura 6.10.

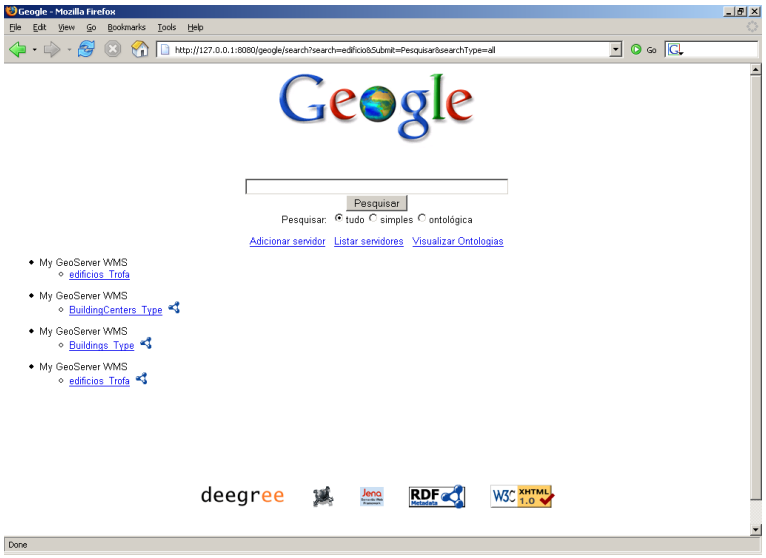


Figura 6.10: Geogle – Exemplo de uma pesquisa com o termo "edifícios"

Como se pode observar, o primeiro resultado é o resultado da pesquisa simples que, como se limita a comparar texto existente nos elementos de cada camada, só conseguiu encontrar um resultado – a camada identificada por edificios_trofa .

Os últimos três resultados, são o resultado da pesquisa ontológica que inferiu que os termos "Building" e "Buildings" referem-se também a edifícios. Isto acontece, porque a base de conhecimento descreve estes conceitos e torna assim possível a relação mais abrangente entre o texto introduzido para pesquisa e as camadas que são efectivamente pesquisadas.

6.3 Geospatial Semantic Web Interoperability Experiment

6.3.1 Introdução

A Semantic Web foi uma ideia introduzida em (Berners-Lee, Hendler, and Lassila, 2001) onde se diz que se trata de "uma nova forma de conteúdos para a Web com significado para os computadores que irão desencadear uma série de novas possibilidades". Em resumo, pretende-se que os dados sejam processados pelos computadores tendo em conta o seu significado.

Para que o significado desses dados esteja presente há que explorar um conjunto de soluções que permitam formalizar esse conhecimento, à luz dos conceitos apresentados no capítulo 4. Neste sentido, o W3C, através dum grupo de trabalho criado para o efeito, *W3C Semantic Web Activity Working Group*[7], tem trabalhado na criação de normas para permitir a formalização do conhecimento, nomeadamente o RDF e a OWL (secção 4.6.1).

A Geospatial Semantic Web pretende estender a Semantic Web para que esta forneça um melhor suporte à IG, ou seja, tendo em conta as suas características especiais (Rocha, 2004) como sejam, a dependência espacial, heterogeneidade e o seu valor relativo (quando mais detalhada está a IG, mais interessa aos habitantes dessa área).

Em Maio de 2005, o consórcio OpenGIS deu início a uma experiência a que chamou *Geospatial Semantic Web Interoperability Experiment*. Formalmente uma *Interoperability Experiment* tem sempre o objectivo geral de, através da colaboração entre diversos membros do consórcio OpenGIS, testar e avaliar determinadas funcionalidades. No caso concreto da GSW IE, pretende-se criar um caso de estudo em que é feito um inquérito semântico vertical, ou seja, que abranja todas as entidades e operações envolvidas desde o cliente até ao serviço, retornando a informação relevante pretendida.

6.3.2 Objectivos

Atingir a interoperacionalidade semântica entre os Geo WS existentes e os seus clientes é o objectivo principal desta iniciativa. Nesta experiência recorre-se aos WFS para testar a extensão e/ou implementação de serviços, que combinados com os mecanismos já existentes para descrição semântica (OWL, OWL-S, etc.), sejam capazes de retornar informação relevante num determinado contexto de utilização.

Para isso identificou-se um caso de uso bem definido sobre o qual se vão concretizar todas as ideias resultantes deste trabalho. Esta experiência é constituída por várias activi-

[7] `http://www.w3.org/2001/sw`

dades (Lieberman, Pehle, and Dean, 2005):

- Desenvolvimento de ontologias.
- Desenvolvimento de interfaces para o WFS que disponibilizem informação do serviço em OWL-S referenciando as ontologias previamente mencionadas.
- Extensão dos interfaces dos serviços para que estes aceitem pedidos feitos numa linguagem capaz de fazer inquéritos semânticos.
- Utilização ou desenvolvimento de ferramentas para gerar IG com informação semântica.
- Implementação de componentes distribuídos para, conjuntamente com os serviços, processarem essa informação semântica.

O caso de uso abordado nesta experiência pretende responder à seguinte questão: "Quais os aeroportos que, numa distância de 500 milhas de Kandahar, suportam o avião C5A?"

Esta questão é susceptível a várias interpretações que originam ambiguidades, por exemplo (Lieberman et al., 2005):

- A "distância" é calculada em linha recta (um círculo com 500Km de raio)?
- Qual o significado de "suportam"? O que é um C5A? Um modelo ou um avião em concreto?
- "Kandahar" é no Afeganistão? Refere-se à região administrativa ou à cidade?

Para que todas estas questões sejam respondidas, utilizando os mecanismos actuais, teria de se reunir informação obtida de diversas fontes com informação geográfica ou não. A análise dessa informação, teria que ser feita por um humano, que faria a selecção dos dados relevantes, interpretava-os e posteriormente chegaria a uma resposta. Por outro lado, utilizando uma abordagem orientada à Semantic Web, a selecção e integração das fontes de informação, a escolha da informação relevante e a análise dessa informação seria feita de forma automática. O utilizador faria apenas a pergunta ao sistema que teria a responsabilidade de coordenar a composição da resposta recorrendo às diversas fontes de informação disponíveis.

O produto final desta iniciativa será uma proposta de extensão das especificações do WFS (OpenGIS, 2002a) e do *Filter Encoding* (OpenGIS, 2005) recorrendo a um conjunto de ontologias de suporte.

6.3.3 Arquitectura

A implementação do caso de uso mencionado anteriormente passa pela definição de uma arquitectura ou estratégia que defina o papel de cada um dos actores presentes neste

processo. Até agora foram identificadas três abordagens arquitecturais, e apesar de ainda se encontrarem em discussão, convém mencionar cada uma delas.

Os principais actores presentes nas três abordagens são: o **cliente**, que efectua um pedido, a **base de conhecimento** (designada também por *Semantic Middelware* pois tem alguma capacidade de processamento em relação a uma simples base de conhecimento), e um conjunto de **WFS**.

Todas as abordagens partilham o facto do cliente fazer um pedido a uma base de conhecimento semântica utilizando uma linguagem apropriada, por exemplo a *Simple Protocol and RDF Query Language* (SPARQL). O papel da base de conhecimento em todas as abordagens é fazer inferências por forma a identificar e construir as questões a submeter aos WFS seleccionados. As respostas podem conter informação semântica (através de um conjunto de instâncias de classes OWL) e informação geográfica codificada em GML.

As abordagens arquitecturais que gozam de maior consenso aparecem sucintamente descritas em (Lieberman et al., 2005) (ver figuras[8] 6.11, 6.11 e 6.11):

Abordagem 1 existem duas formas de interacção entre o cliente e a base de conhecimento:

- descoberta do serviço: o cliente efectua um pedido à base de conhecimento via a sua interface CS-W (*Catalog Service-Web*) numa linguagem com alguma expressividade semântica com o objectivo de obter informações sobre o serviço. A base de conhecimento faz as inferências necessárias por forma a delegar este pedido para um CS-W convencional.
- inquéritos ao serviço descoberto: o cliente faz um pedido à base de conhecimento que retorna um inquérito. Este inquérito corresponde a um pedido GetFeature que é efectuado posteriormente pelo cliente ao WFS.

Abordagem 2 a descoberta do serviço nesta abordagem é idêntica à abordagem anterior. Os pedidos efectuados sobre o serviço descoberto seguem também a mesma linha da abordagem 2. A diferença é o facto do pedido GetFeature não ser feito pelo cliente, mas sim pela base de conhecimento, que processa o GML retornado traduzindo-o para OWL antes de retornar para o cliente.

Abordagem 3 nesta abordagem, o cliente utiliza uma linguagem semântica para efectuar o inquérito (SPARQL). A base de conhecimento traduz esse inquérito num ou mais pedidos GetFeature e encaminha-os aos WFS disponíveis. O GML retornado pelos WFS é transformado para incluir informação semântica, para depois ser retornado ao cliente.

Na primeira abordagem, uma vez que o WFS não interage directamente com a base de conhecimento, esta necessita de ter conhecimento prévio sobre a sua funcionalidade e estrutura para poder fazer as inferências necessárias por forma a compor o GetFeature. Na segunda proposta, a base de conhecimento acede directamente ao WFS, o que simplifica

[8] Estas figuras foram obtidas em `http://portal.opengeospatial.org/wiki/twiki/bin/view/GSWie/WebHome`

a interacção do cliente e elimina a forte dependência estática entre WFS e base de conhecimento. Na terceira abordagem é introduzido uma outra forma de interacção mais expressiva, potenciando ao cliente a capacidade de fazer pedidos mais poderosos.

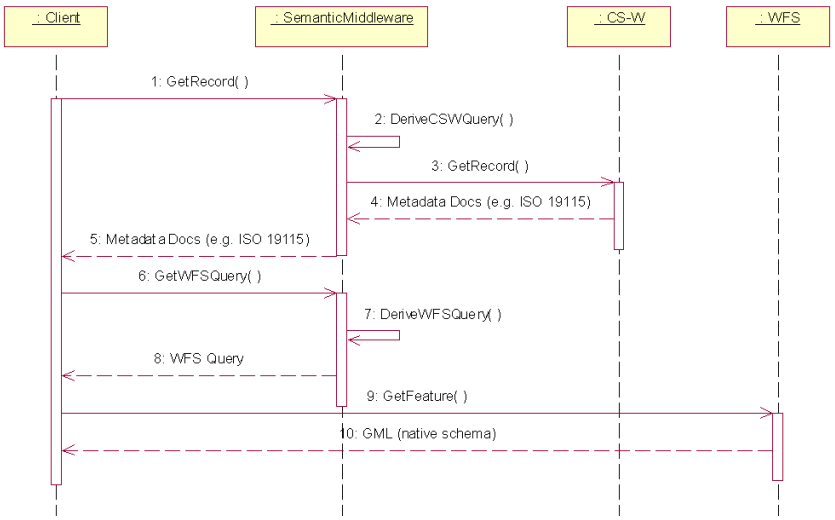


Figura 6.11: Abordagem 1

6.3.4 Ontologias

A OWL foi a linguagem escolhida para o desenvolvimento das ontologias que serão criadas no âmbito da GSW IE. Apesar de não estar clara a arquitectura ontológica que será utilizada, há já algum entendimento sobre esta matéria. Esta arquitectura foi baseada em (Kolas, Hebeler, and Dean, 2005) e prevê a existência ou criação das seguintes ontologias (ver figura 6.14):

- *Domain Ontology*
- *Feature Domain Ontology*
- *Base Geospatial Ontology*
- *Geospatial Filter Ontology*
- *Geospatial Service Ontology*

Domain Ontology

A descrição do conhecimento numa perspectiva orientada ao utilizador é feita através de uma classe de ontologias construídas por diversas comunidades de utilizadores e disponíveis

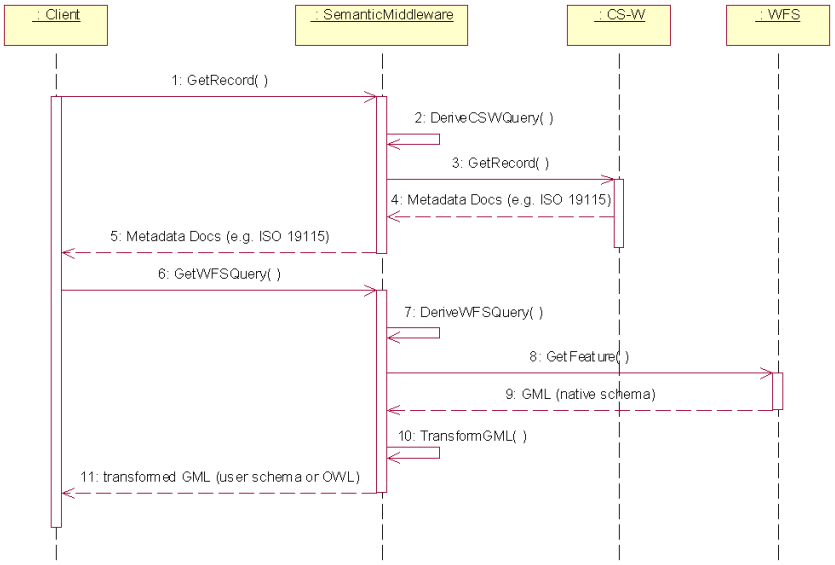


Figura 6.12: Abordagem 2

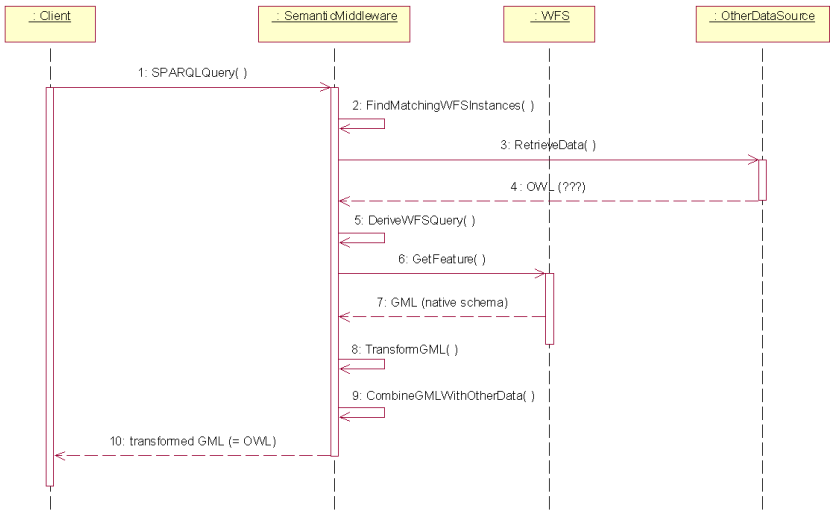


Figura 6.13: Abordagem 3

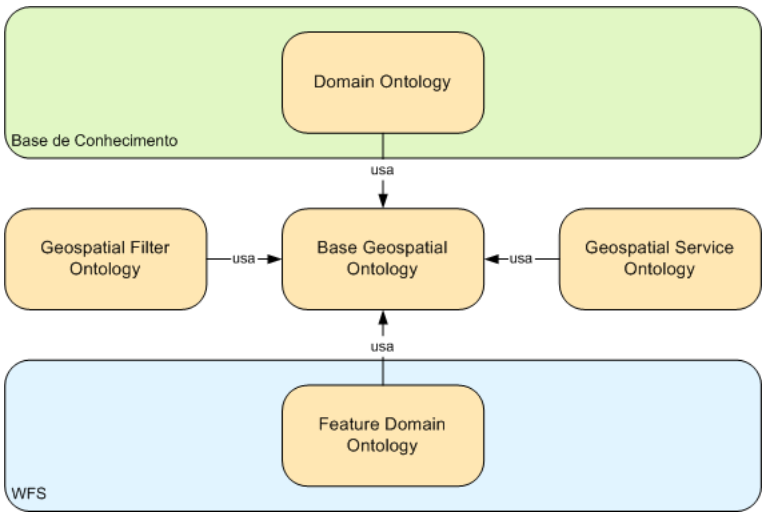


Figura 6.14: Arquitectura de Ontologias

publicamente. Porque o seu principal objectivo é a representação do conhecimento na óptica do utilizador que vai inquirir a base de conhecimento, esta(s) ontologia(s) deve(m) ser capaz(es) de descrever o pedido e os termos que são utilizados no mesmo (eventualmente referenciando outras ontologias).

Esta classe de ontologias utiliza a *Base Geospatial Ontology* para referenciar termos ou conceitos geográficos.

Na listagem 6.2 apresenta-se a ontologia de domínio criada no âmbito da GSW IE, onde se representam os conceitos de *airport* e *C5*.

Listagem 6.2: Feature Domain Ontology (Classes C5 e Airport)

```
157 ...
158   <owl:Class rdf:ID="C5">
159     <rdfs:subClassOf>
160       <owl:Class rdf:ID="Plane"/>
161     </rdfs:subClassOf>
162   </owl:Class>
163
164   <owl:Class rdf:ID="Airport">
165     <rdfs:subClassOf>
166       <owl:Restriction>
167         <owl:onProperty>
168           <owl:ObjectProperty rdf:ID="isLocatedAt"/>
169         </owl:onProperty>
170         <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
171         >1</owl:cardinality>
172       </owl:Restriction>
173     </rdfs:subClassOf>
174     <rdfs:subClassOf>
175       <owl:Restriction>
176         <owl:onProperty>
177           <owl:InverseFunctionalProperty rdf:ID="comprisesRunway"/>
```

```
178           </owl:onProperty>
179           <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
                  int"
180           >1</owl:minCardinality>
181         </owl:Restriction>
182       </rdfs:subClassOf>
183       <rdfs:subClassOf>
184         <owl:Class rdf:ID="ComplexFeature"/>
185       </rdfs:subClassOf>
186     </owl:Class>
187   ...
```

Feature Domain Ontology

Por oposição à classe de ontologias denominadas por *Domain Ontology*, aparece outra classe de ontologias denominada *Feature Domain Ontology* com o propósito de representar o conhecimento ao nível dos serviços.

Esta ontologia, ou classe de ontologias, descreve a estrutura das entidades que serão retornadas pelo WFS. É fortemente dependente da *Base Geospatial Ontology*. Pode fazer-se a analogia com operação describeFeatureType do WFS, que também retorna a estrutura das entidades.

Base Geospatial Ontology

Esta ontologia é horizontal a todas as outras apresentadas nesta secção, pois todas são dependentes desta. Um dos objectivos desta ontologia é dar ao GML outra expressividade incorporando, para isso, os seus conceitos. A listagem 6.3 contém algumas classes e propriedades definidas para a GSW IE.

Pelo facto de alicerçar as ontologias mencionadas nesta secção, é uma forte candidata a ser proposta como norma do OpenGIS.

Listagem 6.3: Base Geospatial Ontology

```
9   ...
10    <owl:Ontology rdf:about=""/>
11    <owl:Class rdf:ID="Geometry"/>
12    <owl:Class rdf:ID="Point">
13      <rdfs:subClassOf rdf:resource="#Geometry"/>
14    </owl:Class>
15
16    <owl:DatatypeProperty rdf:ID="latitude">
17      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
18    </owl:DatatypeProperty>
19
20    <owl:DatatypeProperty rdf:ID="longitude">
21      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
22    </owl:DatatypeProperty>
23  ...
```

Geospatial Filter Ontology

Esta ontologia pretende descrever as relações espaciais de uma forma normalizada. Ao contrário da *Feature Domain Ontology* que é apenas utilizada pelo WFS ou da *Domain Ontology*, que é utilizada apenas pelo utilizador, a *Geospatial Filter Ontology* pretende abranger ambos os contextos.

A formalização de conceitos como "perto", "longe", etc, sob a forma de uma ontologia vai permitir ao utilizador fazer pedidos utilizando esses conceitos em que o seu significado é bem entendido. A listagem 6.4 é uma parte da ontologia deste tipo que foi criada no âmbito da GSW IE.

Listagem 6.4: Geospatial Filter Ontology

```
...
  <owl:Class rdf:ID="Equals">
    <rdfs:subClassOf rdf:resource="#BinarySpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="Disjoint">
    <rdfs:subClassOf rdf:resource="#BinarySpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="Touches">
    <rdfs:subClassOf rdf:resource="#BinarySpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="Within">
    <rdfs:subClassOf rdf:resource="#BinarySpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="Overlaps">
    <rdfs:subClassOf rdf:resource="#BinarySpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="Crosses">
    <rdfs:subClassOf rdf:resource="#BinarySpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="Intersects">
    <rdfs:subClassOf rdf:resource="#BinarySpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="Contains">
    <rdfs:subClassOf rdf:resource="#BinarySpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="DistanceBuffer">
    <rdfs:subClassOf rdf:resource="#SpatialOp"/>
  </owl:Class>
  <owl:Class rdf:ID="DWithin">
    <rdfs:subClassOf rdf:resource="#DistanceBuffer"/>
  </owl:Class>
  <owl:Class rdf:ID="Beyond">
    <rdfs:subClassOf rdf:resource="#DistanceBuffer"/>
  </owl:Class>
...
```

Geospatial Service Ontology

Esta ontologia pretende descrever a interface de um serviço e será codificada em OWL-S. Comparando com a norma actual do WFS, pode dizer-se que se trata de um GetCapabilities semântico, que descreve a sua interface por forma a fornecer informação acerca do fornecedor do serviço, da sua funcionalidade e de outras propriedades que o caracterizem.

Com esta informação, a base de conhecimento consegue decidir se um determinado serviço cumpre ou não os requisitos que são necessários para uma dada operação. Consegue também avaliar se a informação que um serviço retorna pode ser complementada com a de outro serviço (composição de serviços).

Com a apresentação destas duas abordagens que visam possibilitar a descoberta e exploração de serviços de uma forma mais autónoma, cumpre-se um dos objectivos principais deste trabalho de mestrado. Quer uma abordagem, quer outra, podem ser melhoradas. A continuação destes trabalhos é discutida na secção 7.2 do capítulo seguinte.

Capítulo 7

Conclusões e Trabalho Futuro

É neste capítulo que é encerrada esta dissertação recapitulando os objectivos que foram inicialmente propostos e de que forma foram alcançados (na secção 7.1), exploram-se os aspectos a melhorar num trabalho a ser desenvolvido posteriormente, na secção 7.2, e por fim, na secção 7.3, tecem-se alguns comentários, que se julgam oportunos com a conclusão deste trabalho.

7.1 Objectivos Alcançados

O estudo dos XML WS (capítulo 2) permitiu adquirir um conhecimento sobre o que a tecnologia tem para oferecer no tal nível abstracto e agnóstico a domínios específicos de intervenção como é o da IG. Através da utilização de exemplos, demonstrou-se a utilização dos WS e algumas das suas potencialidades no que diz respeito à sua independência de plataforma e linguagem de programação. Os Geo WS, porque antecederam os XML WS não foram desenvolvidos segundo os mesmos parâmetros destes. Mas nem por isso deixam de ter um denominador comum, que é o facto de permitirem um nível abstracção tal que possibilita não só encapsular dados com formatos e proveniências muito diversas por detrás de interfaces comuns (integração de dados), como vieram permitir a composição de serviços dispersos na Web, que podem desta forma contribuir para, em conjunto, oferecer novas funcionalidades (interoperacionalidade). Alguns destes Geo WS, estudados no capítulo 3, permitiram por um lado comprovar a interoperacionalidade sintáctica e por outro, levantar as questões semânticas que se levantaram ao longo da dissertação.

Em (Rocha, 2004) a interoperacionalidade é definida como sendo "a capacidade de comunicar, executar programas ou transferir dados entre diferentes unidades funcionais sem o utilizador se preocupar com as características específicas de cada uma dessas unidades". Conforme foi argumentado, mesmo recorrendo aos Geo WS, o utilizador ainda é actor fundamental para eliminar ambiguidades semânticas.

Também em (Araújo and Rocha, 2004) foram identificados duas possíveis alternativas para que fosse possível a inclusão de meta-informação ao nível do serviço que o descreveria

de forma mais alargada. No capítulo 5, estudou-se essa abordagem que peca pela falta de definição de termos e respectivas relações para o domínio da IG. Ou seja, poder-se-ia questionar o significado da própria meta-informação!

A utilização de ontologias aparece naturalmente como uma resposta aos desafios semânticos enunciados na secção 5.3 que são causados pela existência de ambiguidade sintáctica (Goh, 1997). Na secção 5.5 enumeram-se as principais causas para essa ambiguidade – conflitos denominativos, conflitos de escala e conflitos de equidade – para concluir que é na resolução destes conflitos que reside a solução para se poder ter interoperacionalidade semântica.

Algum do trabalho que tem sido desenvolvido foi descrito na secção 6.1, onde foram enumeradas algumas das iniciativas que se consideraram mais interessantes. Todas elas se pautam pela introdução de mecanismos de descrição semântica dos serviços, das suas *interfaces* e capacidades ((Lemmens and de Vries, 2004), (Hiramatsu and Reitsma, 2004), (Kolas, Hebeler, and Dean, 2005), (Martin et al., 2004)). Nesta dissertação trabalhou-se o nível semântico mas orientado à descrição da IG propriamente dita, para resolução de conflitos denominativos, (Goh, 1997), através da criação de uma camada ontológica e mecanismos de inferência.

A apresentação de um caso de estudo – "Geogle" – descrito na secção 6.2, permitiu estruturar e consubstanciar as questões que se pretendem resolvidas ao longo desta dissertação:

- A tecnologia actualmente permite obter interoperacionalidade sintáctica.
- Há falta de informação semântica associada à IG e aos serviços que a disponibilizam.
- Os mecanismos de descoberta de IG devem partilhar as descrições dos serviços que a disponibilizam.

O Geogle explora estas questões através de funcionalidade que lhe permite:

- Tirar partido da composicionalidade entre serviços, combinando, armazenando e inspeccionando essa informação.
- Demonstrar a interoperacionalidade sintáctica entre Geo WS através de um algoritmo de pesquisa em Geo WS conformes à norma do OpenGIS.
- Resolver os problemas da pesquisa semântica, através da utilização de uma ontologia multi-língua partilhada pelos mecanismos de pesquisa e pela meta-informação dos Geo WS.

A participação no GSW IE veio aprofundar a abordagem à problemática da semântica, pois o âmbito desta experimentação é mais lato do que o considerado no Geogle. Enquanto que no Geogle é aceitável apenas uma ontologia comum de entidades geográficas e uma camada de mediação e inferência que processa a informação com base nessa ontologia, na GSW IE são necessários vários níveis semânticos que começam na questão formulada até ao

tratamento de informação proveniente de diferentes serviços, justificando a criação de vários níveis ontológicos e ferramentas e mecanismos de integração com as normas existentes.

Também foi muito pedagógico participar nestas discussões com outros membros do consórcio OpenGIS, com mais experiência nesta área. Todo o percurso desta IE, que irá resultar na criação ou adaptação de normas do OpenGIS, pode vir a criar oportunidades de trabalho e investigação a explorar no futuro.

Paralelamente, esta dissertação, para além das questões de âmbito geográfico, e de certa forma como *efeito secundário*, contribuiu directa ou indirectamente nas seguintes vertentes:

Adopção de formatos abertos ao utilizar protocolos e formatos normalizados está claramente a contribuir-se para a massificação da utilização dos mesmos e foi esse um dos desafios que se tentou atingir ao longo deste trabalho. Um dos requisitos para a interoperacionalidade é o entendimento comum entre os diversos actores que só poderá ser obtido através da normalização.

Utilização de software aberto todos os componentes de desenvolvimento directamente relacionados com o trabalho que foi feito e agora termina foram escolhidos por forma a privilegiar aqueles que são fornecidos de forma gratuita e sem restrições. O projecto deegree é um bom exemplo, em que se contribuiu para a detecção e resolução de problemas. Recorrer a projectos desta índole é contribuir para a sua utilização e divulgação, em detrimento de outras soluções potencialmente caras e proprietárias.

Web Services estudar a tecnologia dos WS permitiu que fossem criados os alicerces para uma futura integração com Geo WS baseados em SOAP. Ao utilizar esta tecnologia contribui-se para potenciar a evolução e desenvolvimento de soluções abertas, pelo que o capítulo 2 contribui de forma pedagógica e exemplificativa para a sua utilização em outros domínios.

Geo WS imaginar um mundo em que a IG pode ser partilhada livremente e sem restrições de ordem tecnológica é um dos objectivos do consórcio OpenGIS. Ao recorrer a Geo WS como base para todo este trabalho contribuiu-se para a sua divulgação e utilização.

7.2 Trabalho Futuro

Em termos conceptuais existem diversos tópicos que poderiam ser objecto de estudo em futuros trabalhos. Em termos mais abstractos a participação em projectos com o objectivo lato de promover massificação da IG e interoperacionalidade entre os sistemas que a suportam vai continuar a ser, sem qualquer dúvida, uma área em que investiremos.

7.2.1 Ontologias

A utilização de mecanismos baseados em ontologias está ainda em fase de estudo por parte do consórcio OpenGIS e é ainda um tópico muito quente no âmbito dos SIG. O acompanhamento das iniciativas que daqui apareçam e a participação nas mesmas é um contributo que gostaríamos de dar.

Ainda numa perspectiva de introdução de ontologias para resolver problemas semânticos, gostaríamos de explorar a resolução de conflitos de escala e conflitos de equidade, para assim se alargar o campo de acção da interoperacionalidade.

O ser humano ainda é fundamental em toda esta cadeia? A resposta é sim, mesmo com a interoperacionalidade semântica atingida, ainda é o humano que escreve/desenvolve as ontologias. Explorar soluções de extracção de ontologias de Geo WS pode ser um dos próximos passos na procura por sistemas que "realmente" gozem da interoperacionalidade.

7.2.2 Geogle

O Geogle que começou como um caso de estudo tem potencial para crescer, não só como aplicação independente, mas também como WS ou plataforma de desenvolvimento de software.

Gostaríamos de continuar este projecto, possivelmente partilhando o seu código fonte, e melhorar a sua funcionalidade em diversos aspectos:

Robô de pesquisa actualmente o Geogle mantém os servidores a pesquisar manualmente, isto é, essa gestão é da responsabilidade do administrador. A pesquisa e manutenção de novos Geo WS poderia ser confinada a um robô, o que iria aumentar a eficiência e abrangência do motor de busca.

Resolução de conflitos de escala e equidade a resolução destes tipos de conflitos iria ser um passo em frente no retorno de resultados.

Criação automática de ontologias em paralelo com o motor de busca, a geração automática de ontologias iria reduzir substancialmente a intervenção do ser humano, que ainda tem a responsabilidade de as manter.

Integração com XML Web Services adaptar o Geogle para ser um XML Web Service seria uma mais valia importante na medida do que se tem reiterado ao longo desta dissertação, reforçando a utilização de protocolos abertos e permitindo que de forma normalizada, o Geogle pudesse ser acedido.

Estender o Geogle para fazer inferências geográficas utilizar a geografia, para fazer inferências não existentes/explícitas na ontologia. Por exemplo, recorrendo a um *Gazetteer*, pode obter-se os limites de Braga, que podem posteriormente ser utilizados para, analisando os resultados da resposta a um inquérito, associar a localização

ao resultado. Identificada essa associação, poderia trabalhar-se a criação automática/assistida de ontologias ou como mais um mecanismo de busca para aumentar as capacidades do Geogle.

7.3 Considerações Finais

O culminar desta dissertação, em primeiro lugar, representa o recompensar de todo um esforço de trabalho que foi sendo realizado e, não menos importante, em segundo lugar, representa também o começo de uma nova etapa. Etapa essa que, parafraseando um "super herói" da actualidade, significa a partida para "o infinito e mais além".

Há a convicção de que este trabalho abriu os nossos horizontes e que será o ponto de partida para novos estudos e contributos que irão coadjuvar com todos aqueles que querem um mundo onde a IG é partilhada livremente num espírito cooperante e sem limites de nenhuma ordem.

A complexidade desta temática motiva o desenvolvimento e aparecimento de soluções tecnológicas sofisticadas. Por oposição a este emaranhado de soluções emergentes, está a trabalhar-se para simplificar aquele que mais pode beneficiar da utilização transparente da IG – o utilizador comum.

Esperamos que este trabalho tenha contribuído para sofisticar os mecanismos de descrição semântica, para oferecer ao utilizador informação geográfica cada vez mais adequada aos seus propósitos, sem lhe mostrar a complexidade das tecnologias de suporte, ou a dificuldade em criar e articular ontologias com diferentes propósitos.

Bibliografia

[Aguiar et al.1998] Aguiar, Ademar, Alexandre Valente Sousa, Artur Rocha, Aurélio Pires, João Paulo Hespanha, Lígia Silva, and Luís Galiza. 1998. Simat – relatório da especificação de requisitos. Technical report, Instituto de Engenharia de Sistemas e Computadores, Outubro. `http://www.inesc.pt/SIMAT`.

[Ambler2004] Ambler, Scoot. 2004. *The Object Primer – Agile Model-Driven Development with UML 2.0.* Ronin International, 3 edition, Junho. ISBN 0521540186.

[Apache2004a] Apache. 2004a. Apache soap v2.3.1 documentation. `http://ws.apache.org/soap`.

[Apache2004b] Apache. 2004b. Axis user's guide. `http://ws.apache.org/axis/java/user-guide.html`. Version 1.2.

[Apache2004c] Apache. 2004c. The tomcat 5 servlet/jsp container documentation. `http://jacarta.apache.org`.

[Araújo and Rocha2004] Araújo, Mário André and Jorge Gustavo Rocha. 2004. Web services na informação geográfica. In *XML: Aplicações e Tecnologias Associadas*, pages 88–100, Lisboa, Fevereiro. Faculdade de Engenharia da Universidade do Porto. ISBN 972-99166-0-8.

[Ballinger2003] Ballinger, Keith. 2003. Basic profile version 1.0a. `http://www.ws-i.org/ProfilesBasic/2003-08/BasicProfile-1.0a.HTML`, Agosto.

[Bechhofer et al.2001] Bechhofer, Sean, Ian Horrocks, Carole Goble, and Robert Stevens. 2001. OilEd: A reason-able ontology editor for the semantic Web. *Lecture Notes in Computer Science*, 2174:396–405.

[Berners-Lee, Hendler, and Lassila2001] Berners-Lee, Tim, James Hendler, and Ora Lassila. 2001. The semantic web. `http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21`.

[Borst1997] Borst, Willem Nico. 1997. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse.* Ph.D. thesis, Dutch Graduate School for Information and Knowledge Systems.

[Bray, Hollander, and Layman1999] Bray, Tim, Dave Hollander, and Andrew Layman. 1999. Namespaces in xml. `http://www.w3.org/TR/REC-xml-names`.

[Bryan et al.2002] Bryan, Douglas, Vadim Draluk, Dave Ehnebuske, Tom Glover, Andrew Hately, and Yin Leng Husband. 2002. Uddi version 2.04 api specification. `http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm`, Julho. UDDI Committee Specification.

[Buehler et al.2002] Buehler, Kurt, Sam Bacharach, Carl Reed, Cliff Kottman, Chuck Heazel, John Davidson, Yaser Bisher, Harry Niedzwiadek, and John Evans. 2002. Opengis reference model. `http://www.opengis.org/info/orm`, Abril.

[Cerami2002] Cerami, Ethan. 2002. *Web Services Essencials*. O'Reilly, 1 edition, Fevereiro. ISBN 0-596-00224-6.

[Chappell and Jewell2002] Chappell, David A. and Tyller Jewell. 2002. *Java Web Services*. O'Reilly, 1 edition, Março. ISBN 0-596-00269-6.

[Chen, Finin, and Joshi2004a] Chen, Harry, Tim Finin, and Anupam Joshi. 2004a. A context broker for building smart meeting rooms. `http://ebiquity.umbc.edu/_file_directory_/papers/78.pdf`.

[Chen, Finin, and Joshi2004b] Chen, Harry, Tim Finin, and Anupam Joshi. 2004b. An ontology for context-aware pervasive computing environments. `http://ebiquity.umbc.edu/_file_directory_/papers/63.pdf`.

[Chen et al.2004] Chen, Harry, Filip Perich, Tim Finin, and Anupam Joshi. 2004. Soupa: Standard ontology for ubiquitous and pervasive applications. `http://ebiquity.umbc.edu/_file_directory_/papers/105.pdf`.

[Christensen et al.2001] Christensen, Erik, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. 2001. Web services description language (wsdl) 1.1. `http://www.w3.org/TR/wsdl`, Março.

[Chung2003] Chung, Jen Yao. 2003. Web-services computing: Advancing software interoperability, Outubro.

[Clark, Hiramatsu, and Reitsma2004] Clark, Kendall, Kaoru Hiramatsu, and Femke Reitsma. 2004. geoontologies. `http://www.mindswap.org/2004/geo/geoOntologies.shtml`.

[de Almeida1997] de Almeida, José João Dias. 1997. *Dicionários dinâmicos multi-fonte*. Ph.D. thesis, Universidade do Minho.

[Fallside2002] Fallside, David C. 2002. Xml schema part 0: Primer. `http://www.w3.org/TR/xmlschema-0`.

[Farquhar, Fikes, and Rice1996] Farquhar, A., R. Fikes, and J. Rice. 1996. The ontolingua server: A tool for collaborative ontology construction. `citeseer.ist.psu.edu/farquhar96ontolingua.html`.

[Gamma et al.1994] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Professional, 1 edition, Outubro. ISBN 0201633612.

[GETTY2005] GETTY. 2005. About the tgn. http://www.getty.edu/research/conducting_research/vocabularies/tgn/about.html.

[Goh1997] Goh, Cheng Hian. 1997. *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems.* Ph.D. thesis, Massachusetts Institute of Technology, Janeiro.

[Grosso et al.1999] Grosso, W. E., Henrik Eriksson, Ray W. Fergerson, John H. Gennari, Samson W. Tu, and Mark A. Musen. 1999. Knowledge modeling at the millennium (the design and evolution of protégé-2000). In *Proceedings of KAW99*, Canada, Outubro. Springer-Verlag.

[Gruber1993] Gruber, T. R. 1993. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands. Kluwer Academic Publishers. citeseer.ist.psu.edu/gruber93toward.html.

[Guarino1998] Guarino, N. 1998. Formal ontology and information systems.

[Hiramatsu and Reitsma2004] Hiramatsu, Kaoru and Femke Reitsma. 2004. Georeferencing the semantic web: ontology based markup of geographically referenced information. http://www.mindswap.org/2004/geo/geoStuff_files/HiramatsuReitsma04GeoRef.PDF, Abril. Joint EuroSDR/EuroGeographics workshop.

[ISO1986] ISO. 1986. Documentation – guidelines for the establishment and development of monolingual thesauri. Technical report, International Organization for Standardization, Novembro.

[Kolas, Hebeler, and Dean2005] Kolas, Dave, John Hebeler, and Mike Dean. 2005. Geospatial semantic web: Architecture of ontologies, Junho.

[Lemmens and de Vries2004] Lemmens, R.L.G. and M. de Vries. 2004. Semantic description of location based web services using an extensible location ontology. In *Proceedings of Münster GI-days*, Julho. http://www.itc.nl/library/Papers_2004/peer_conf/lemmens_devries.pdf.

[Librelotto, Ramalho, and Henriques2003] Librelotto, Giovani, José Carlos Ramalho, and Pedro Rangel Henriques. 2003. Tm-builder: Um construtor de ontologias baseado em topic maps. http://www.di.uminho.pt/~jcr/XML/publicacoes/artigos/2003/clei.pdf.

[Librelotto2005] Librelotto, Giovani Rubert. 2005. *Topic Maps: da Sintaxe à Semântica.* Ph.D. thesis, Universidade do Minho.

[Lieberman, Pehle, and Dean2005] Lieberman, Josh, Todd Pehle, and Mike Dean. 2005. Semantic evolution of geospatial web services. http://www.w3.org/2005/04/FSWS/Submissions/48/GSWS_Position_Paper.html.

[Lieberman et al.2005] Lieberman, Joshua, Dave Kolas, Todd Pehle, and Peisheng Zhao. 2005. Geospatial semantic web interoperability experiment collaboration space. `http://portal.opengeospatial.org/wiki/twiki/bin/view/GSWie/WebHome`.

[Lopes2003] Lopes, Carlos. 2003. *Web Services: Aplicações Distribuídas sobre Protocolos Internet.* Ph.D. thesis, Universidade do Minho.

[Martin et al.2004] Martin, David, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, and Katia Sycara. 2004. Bringing semantics to web services: The owl-s approach. In *First International Workshop on Semantic Web Services and Web Process Composition*, Julho. San Diego, California, USA.

[Mitra2003] Mitra, Nilo. 2003. Soap version 1.2 part 0: Primer. `http://www.w3.org/TR/soap12-part0`, Junho.

[Nagappan, Skoczylas, and Sriganesh2003] Nagappan, Ramesh, Robert Skoczylas, and Rima Sriganesh. 2003. *Developing Java Web Services.* Wiley Pbulshing, Inc., 1 edition. ISBN 0-471-23640-3.

[OASIS2004] OASIS. 2004. Introduction to uddi: Important features and functional concepts. `http://www.uddi.org`, Outubro.

[OpenGIS1998] OpenGIS. 1998. *The OpenGIS Guide.* Kurt Buehler and Lance McKee, 3 edition, Junho.

[OpenGIS2002a] OpenGIS. 2002a. Web feature service implementation specification.

[OpenGIS2002b] OpenGIS. 2002b. Web map service. `http://www.opengis.org/docs/01-068r3.pdf`.

[OpenGIS2005] OpenGIS. 2005. Opengis filter encoding implementation specification. `http://portal.opengeospatial.org/files/?artifact_id=8340`.

[Ramalho and Henriques2002] Ramalho, José Carlos and Pedro Rangel Henriques. 2002. *XML \& XSL, da teoria à práctica.* FCA. ISBN 972-722-347-8.

[Raskin2004] Raskin, Rob. 2004. Guide to sweet ontologies. `http://sweet.jpl.nasa.gov/guide.doc`.

[Rocha2004] Rocha, Jorge Gustavo. 2004. *Informação Geográfica: Meta-Informação, Codificação e Visualização.* Ph.D. thesis, Universidade do Minho.

[Sabbouh et al.2001] Sabbouh, Marwan, Stu Jolly, Dock Allen, Paul Silvey, and Paul Denning. 2001. Workshop on web services. `http://www.w3.org/2001/03/WSWS-popa/paper08`.

[Saias2003] Saias, José Miguel Gomes. 2003. Uma metodologia para a construção automática de ontologias e a sua aplicação em sistemas de recuperação de informação. Master's thesis, Universidade de Évora, Novembro.

[Smith and Mark1998] Smith, Barry and David M. Mark. 1998. Ontology and geographic kinds. In *Proceedings, International Symposium on Spatial Data Handling (SDH'98)*, pages 308–320, Vancouver, Canada, Julho. `http://www.geog.buffalo.edu/ncgia/i21/SDH98.html`.

[Stelting and Maassen2002] Stelting, Stephen and Olav Maassen. 2002. *Applied Java Patterns*. Sun. ISBN 0-13-093538-7.

[W3C1997] W3C. 1997. Xml – extensible markup language. `http://www.w3.org/XML`.

[W3C2001] W3C. 2001. The semantic web. `http://www.w3.org/2001/sw`.

[W3C2004a] W3C. 2004a. Owl web ontology language – use cases and requirements. `http://www.w3.org/TR/webont-req`.

[W3C2004b] W3C. 2004b. Owl web ontology language overview. `http://www.w3.org/TR/2004/REC-owl-features-20040210`.

[Wache et al.2001] Wache, H., V. ogele, T. Visser, U. Stuckenschmidt, H. Schuster, G. Neumann, and H. ubner. 2001. Ontology-based integration of information - a survey of existing approaches. `citeseer.ist.psu.edu/article/wache01ontologybased.html`.

[Weissenberg and Gartmann2003] Weissenberg, Norbert and Ruediger Gartmann. 2003. Ontology architecture for semantic geo services for olympia 2008. `http://www.gi-tage.de/2003/downloads/gitage2003/tagungsband/weissenberg.pdf`.

[Wiederhold1994] Wiederhold, Gio. 1994. Interoperation, mediation and ontologies. In *Int. Symposium on 5th Generation Computer Systems; Workshop on Heterogeneous Cooperative Knowledge-Bases*, volume W3, pages 33–48, Tokyo, Japan. `citeseer.ist.psu.edu/wiederhold94interoperation.html`.

[Wiederhold and Jannink1999] Wiederhold, Gio and Jan Jannink. 1999. Composing diverse ontologies. In *DS-8*, volume 138, pages 33–48, Rotorua, New Zealand. `http://www-db.stanford.edu/SKC/publications/ifip99.html`.